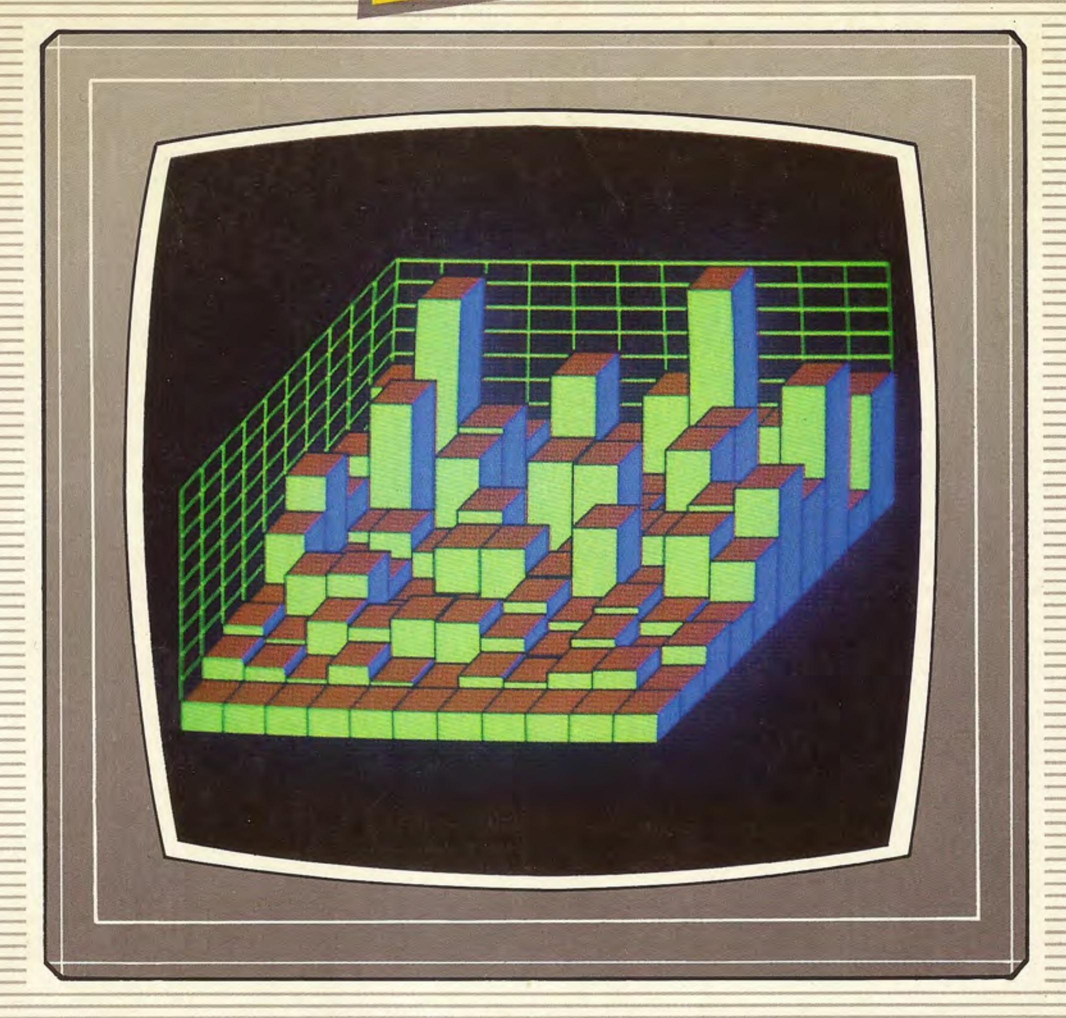
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 2

PASO A PASO



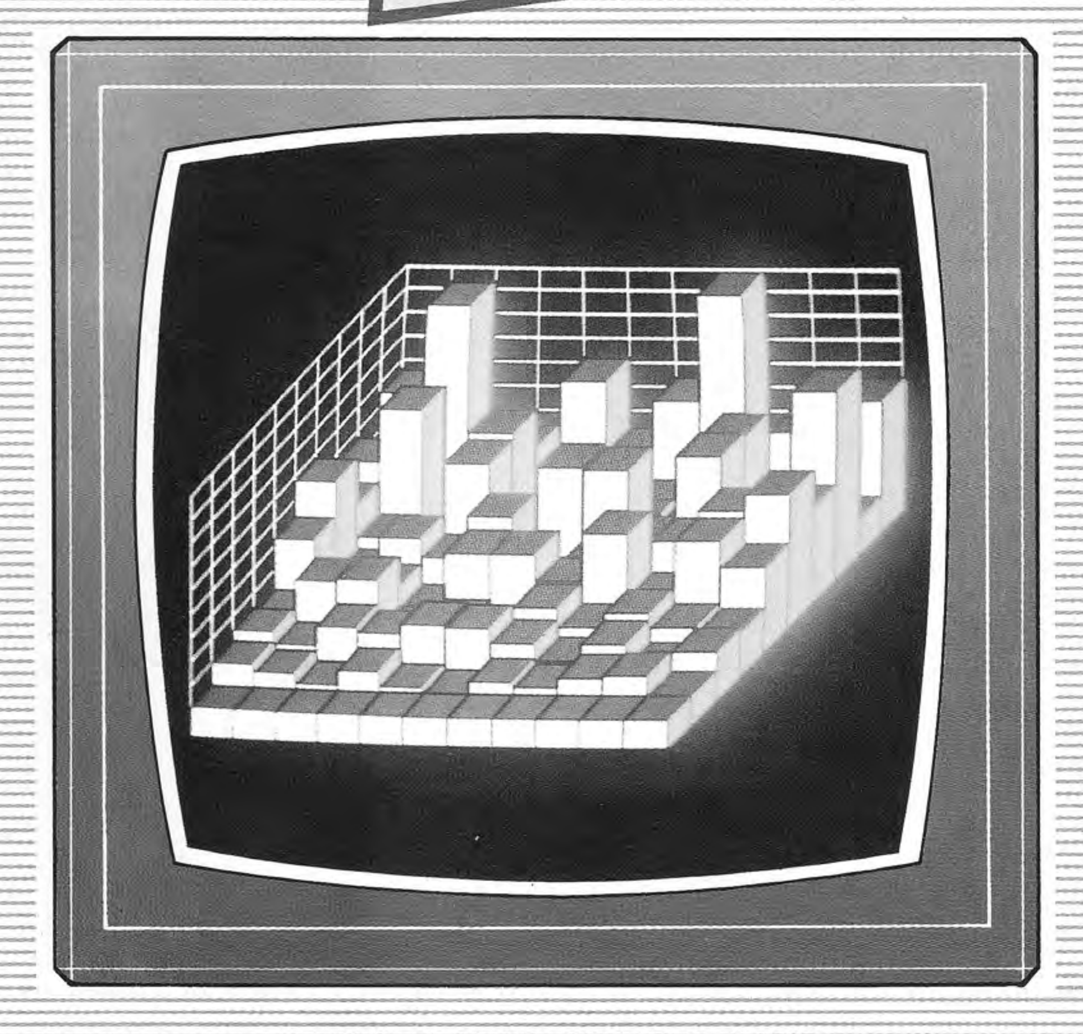
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informatica y programacion 2

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Todo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: Jesús Bocho, Licenciado en Informática. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Fernando Suero, Diplomado en Telecomunicación. OTROS LENGUAJES (Sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8-2.ª planta. Teléf.: 810 52 13. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

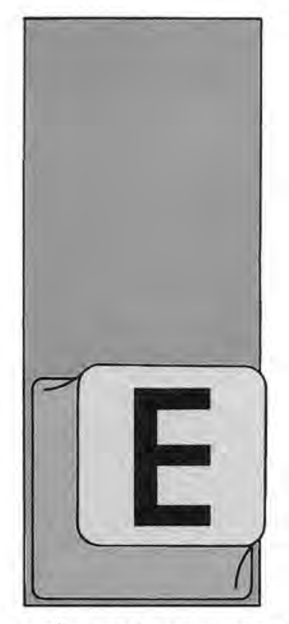
ISBN del tomo: 84-7688-070-7
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8-2.ª planta. Teléf.: 810 52 13. 28020 Madrid.
Abril, 1987.
P.V.P. Canarias: 335,–.

INDICE

INFORMATICA BASICA

ARQUITECTURA DE LOS ORDENADORES



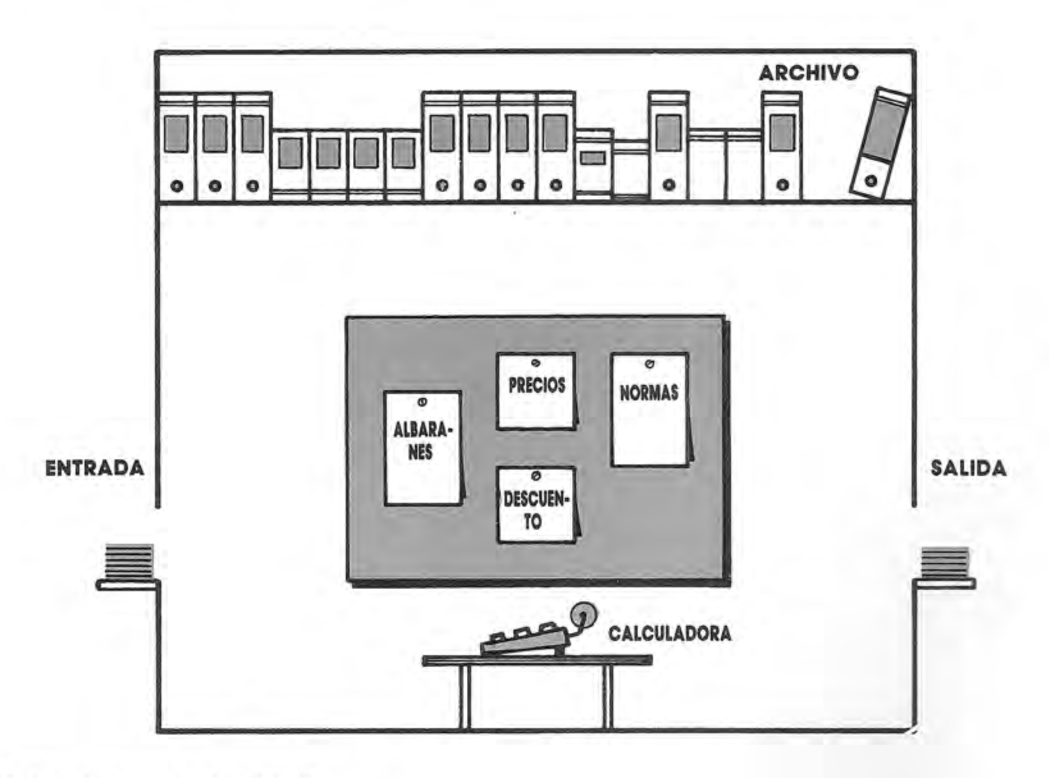
EN este tema vamos a intentar conocer cómo es por dentro un ordenador, punto de trascendencia fundamental para el posterior desarrollo de cualquier aplicación obteniendo su máximo rendimiento.

Fundamentalmente un ordenador está compuesto por dos partes:

— **La unidad central de proceso,** que abreviadamente se conoce por las siglas CPU o UCP, según se utilicen las inglesas o las castellanas. Esta parte es la que constituye el verdadero "centro de trabajo" del ordenador, ya que en ella se ejecutan los programas.

— **Las unidades periféricas,** que son dispositivos que tiene el ordenador para comunicarse con su entorno, los medios por los que se proporciona información al ordenador y aquéllos por los que el ordenador suministra los resultados de los programas. También suelen considerarse como unidades periféricas los distintos dispositivos de almacenamiento secundario de los cuales se hablará más adelante.

Profundizando un poco más en el interior del ordenador veamos de qué partes está compuesta la Unidad Central de Proceso. Para ello vamos a hacer una analogía entre el funcionamiento del ordenador y el de una oficina en la que se ejecutan tareas administrativas (ver figura 1).



Oficina de tareas administrativas.

La oficina de la que hablamos está situada en una habitación en la que sólo hay un empleado. El trabajo de cada mañana le llega por una ventanilla de recepción por la que recibe diariamente los albaranes que contienen una lista de pedidos. El tratamiento que se da a cada albarán dependerá de ciertas normas que le han sido previamente detalladas "programando" las operaciones a realizar. Como útiles de trabajo, el empleado dispone de un tablero en el que va colocando las normas a seguir en cada caso, la lista de precios para cada cliente y la lista de descuentos, y, para las operaciones aritméticas que tenga que hacer, una calculadora.

En el proceso de facturación el empleado realiza los siguientes pasos:

— Lee cuidadosamente la lista de instrucciones que figuran en el tablero, ejecutando una detrás de otra todas las instrucciones.

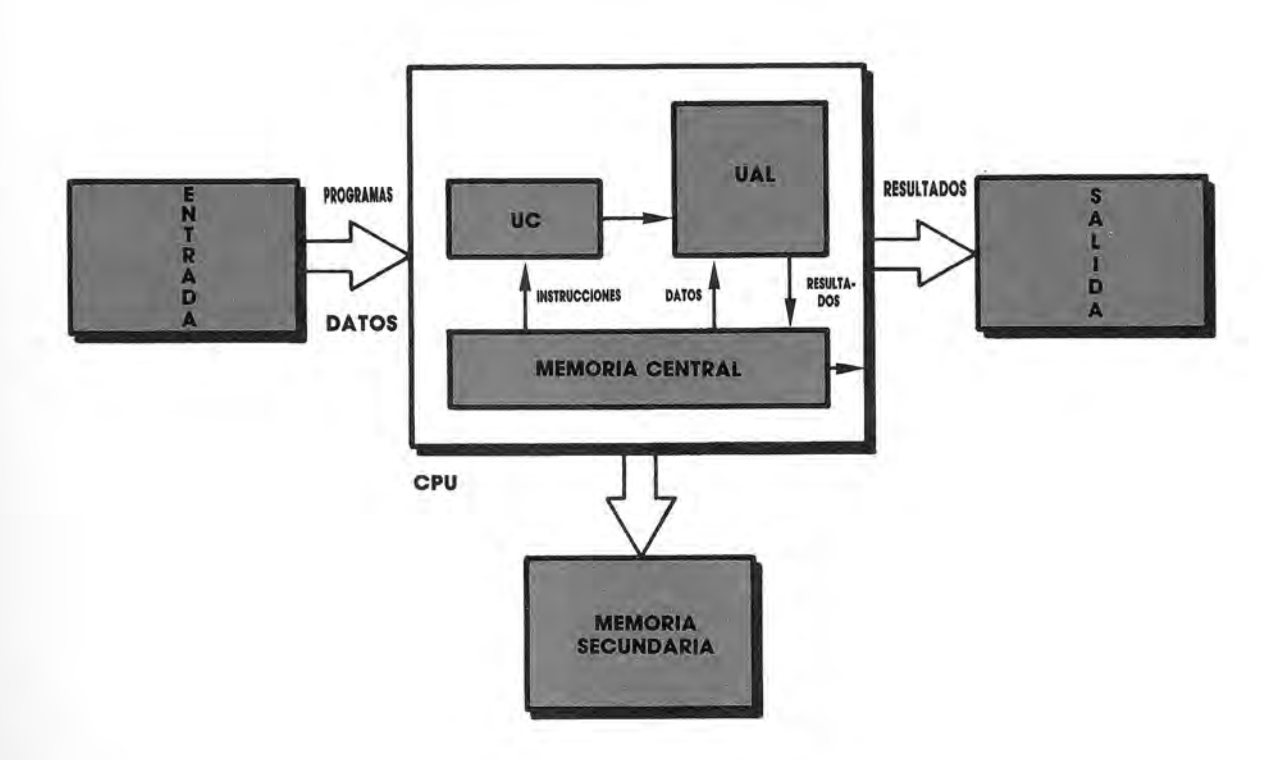
— Con las normas a seguir y los datos necesarios para realizar la facturación realiza las operaciones aritméticas de cálculo de precios y descuento.

— Con los datos y los resultados obtenidos rellena el documento de facturación y lo deposita en la ventanilla de salida.

Estas operaciones se repiten tantas veces como haga falta hasta teminar con el último albarán y su factura correspondiente.

Cuando el volumen de trabajo sea tan grande que ocupe todo el espacio del tablero será necesario que el empleado vaya guardando los documentos en alguna parte. Para ellos dispone de un archivo fuera de la oficina en una habitación separada.

Una vez visto este ejemplo de trabajo será más fácil entender el funcionamiento y las distintas partes del ordenador. En la figura 2 vemos representados los distintos elementos:



Partes de un ordenador.

— **Los dispositivos de entrada/salida** del ordenador corresponderían en nuestro ejemplo a las dos ventanillas que comunican la oficina con el exterior. Por la que le llegan al empleado los albaranes sería "el dispositivo de entrada de datos", y por la que él va entregando las facturas actualizadas sería "el periférico de salida" del ordenador.

— El director de la empresa, encargado de determinar las órdenes que tiene que ejecutar el empleado a la hora de calcular las facturas, y controlar si dicha ejecución ha sido bien realizada, es el equivalente al elemento U.C. o unidad de control.

— La pequeña calculadora utilizada para los cálculos es lo que en términos de arquitectura se denomina unidad aritmético-lógica.

— El tablero en el que se van colocando tanto las acciones a seguir como los datos que van siendo obtenidos es el elemento M.C., o memoria central donde se almacenan los programas y datos. Esta memoria tiene normalmente una capacidad finita, insuficiente para almacenar todos los datos.

— Cuando los documentos no cabían en la oficina se almacenaban fuera de ella. En el caso del ordenador éste almacenamiento secundario es lo que constituye la memoria secundaria y serán normalmente discos magnéticos, cintas, etcétera.

En resumen, los órganos básicos de un ordenador son:

— La unidad central de proceso, compuesta por:

- La unidad de control.
- La unidad aritmético-lógica.
- La memoria central.

— Los dispositivos periféricos:

- De entrada/salida.
- De almacenamiento secundario.

Una vez conocido el "interior" de un ordenador vamos a ver cómo funciona más detalladamente cada una de sus partes.



Unidad central de proceso

La unidad de control

Como vimos anteriormente, la U.C. es el órgano "más inteligente" del ordenador, que se encarga de dirigir y controlar cada una de las acciones que se ejecutan en él. Es decir:

— Analiza e interpreta las instrucciones del programa que se está ejecutando.

— Dirige el funcionamiento de las distintas partes del ordenador mediante órdenes dirigidas a las mismas controlando su actuación.

Para saber qué operaciones se tienen que realizar se habrá cargado previamente en la memoria del ordenador el correspondiente programa. Una vez hecho esto, los pasos que sigue la U.C. son los siguientes:

— Extrae de la memoria la instrucción a ejecutar. Para ello existe en la máquina un "apuntador" que señala cuál es la instrucción siguiente. Este "apuntador" es un registro llamado contador de programa, que contiene la dirección de memoria donde se encuentra esta instrucción. Cuando se extrae, el P.C. (abreviatura de siglas inglesas) se incrementa para apuntar a la dirección de la siguiente instrucción del programa.

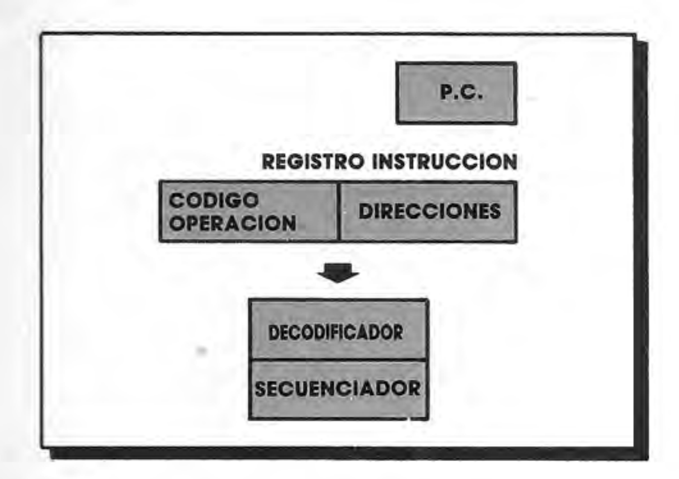
— Una vez extraída se coloca en el llamado registro de instrucción que consta de dos partes: una, que contiene el código de operación que identifica la operación a ejecutar (suma, resta...) y la segunda, que contiene la dirección en la que está almacenado el operando.

— Si la operación identificada es de tipo aritmético se ordena a la U.A.L. que la realice.

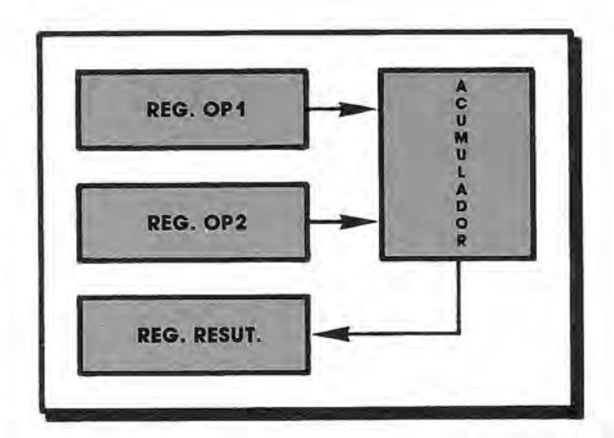
— Si la operación ha proporcionado nuevos datos, éstos se almacenan en la memoria principal (ver figura 3).

La unidad aritmética lógica

La U.A.L. es la parte del ordenador encargada de efectuar las operaciones que indiquen las instrucciones del progra-




Partes de la unidad de control.




Unidad aritmético-lógica.

ma. Las operaciones que puede realizar esta unidad son de tipo:

— Aritmético: suma, resta, multiplicación, división.
— Lógicas: comparación, negación...

Estas operaciones se realizan sobre datos o informaciones que previamente han sido almacenados en memoria, y nunca podrán efectuarse sobre más de dos operandos, de manera que si hay que hacer la suma de tres cantidades será necesario sumar de dos en dos, teniendo, por tanto, dos sumas.

Los pasos a seguir en la U.A.L. para realizar operaciones son:

— Cargar el primer operando en el acumulador (registro de almacenamiento temporal).
— Realizar la operación con el segundo operando y el contenido del acumulador.

Cargar el contenido del acumulador en la dirección del resultado.

Para poder hacer todo esto se debe proporcionar a la U.A.L. los siguientes datos:

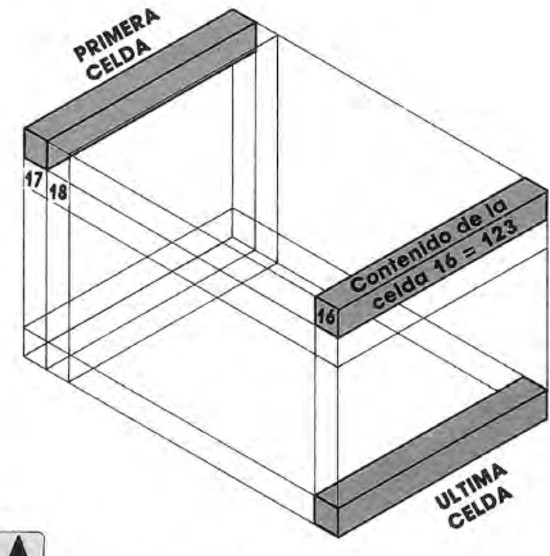
— Código de operación, que indique la operación a efectuar.
— Dirección de la célula en la que se encuentra almacenado el primer operando.
— Dirección del segundo operando.
— Dirección de la célula en la que se almacenará el resultado.

Podemos ver más detalladamente el funcionamiento de la unidad aritmético lógica en la figura 4.

Memoria central

La memoria central del ordenador es el elemento que se encarga de almacenar datos y programas (información en general). Por tanto, la memoria puede considerarse como un órgano pasivo en el que se:

— Introduce información ("escribir en memoria").
— Extrae información ("leer en memoria").




Memoria central.

La memoria está constituida por un conjunto de celdas ordenadas numéricamente por sus direcciones y que son capaces de albergar un dato o una instrucción. Para poder acceder al contenido de una celda de memoria o escribir

algo en ella la memoria dispone de dos registros:

— Registro de dirección de memoria: contiene en un momento dado la dirección de la celda que se trata de buscar.

— Registro de información de memoria: en él deposita la memoria el contenido de la celda seleccionada en el caso de una operación de lectura, o bien contiene una información para que sea depositada en la celda seleccionada en el caso de escritura.

Según vaya a efectuarse una operación de lectura o de escritura, se seguirán los siguientes pasos:

Lectura:

— Almacenar la dirección de la celda en la que se encuentra la información en el registro de dirección.

— Cargar en el registro de intercambio la información contenida en la celda apuntada por el registro de dirección.

— Transferir el contenido al registro de la CPU que corresponda.

Escritura:

— Transferir al registro de intercambio la información.

— Almacenar la dirección de la celda receptora de información.

— Cargar el contenido de intercambio en la celda apuntada por el registro de dirección.

Una propiedad de la memoria central es que cuando se "grabe" en una celda la información, automáticamente queda destruida la información que contuviese anteriormente. No ocurre esto, evidentemente, al leer el contenido de una celda, que consiste únicamente en hacer un duplicado de la información que seguirá permaneciendo en dicha celda.



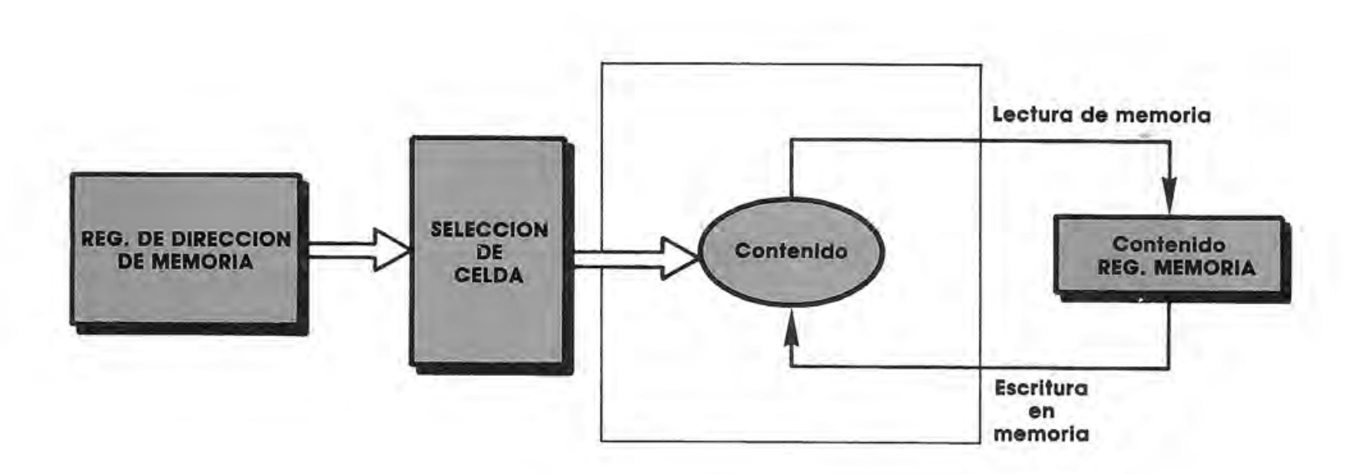
Dispositivos periféricos

Podemos hacer una clasificación en dos grandes grupos:

— Los que se encargan del intercambio de información entre el ordenador y el exterior; y

— Los dispositivos de almacenamiento de grandes volúmenes de información.

Dentro del primer grupo podemos distinguir los que se encargan de introducir información al ordenador, que serían los de entrada. De salida serían aquéllos por los cuales el ordenador proporciona información. Normalmente la información que introducimos al ordenador no es directamente legible por él, por lo que hay que "transcribir" dicha información a un sistema legible por él (es decir, ceros y



Proceso de escritura/lectura en memoria.

unos) mediante la función de codificación; análogamente, será necesario realizar la operación contraria, llamada decodificación para que los resultados de los programas ya ejecutados sean inteligibles al hombre.

Como dispositivos de este tipo podemos citar, como más conocidos:

— Teclado: dispositivo de entrada.
— Pantalla, impresora: dispositivos de salida.

En cuanto al segundo grupo mencionado, también llamado "memoria externa", se encarga de almacenar los datos del ordenador que necesitará recuperar en cualquier momento y que en un instante dado no cupieron en la memoria interna del ordenador.

Dentro de este grupo están los conocidos discos, disquetes, cintas magnéticas, etcétera.

MAQUINA 6502
(COMMODORE 64)

INTRODUCCION



Si tuviéramos que definir el microprocesador 6502 de forma rápida y sencilla,diríamos que se trata de una CPU integrada en una sola "pastilla". Su esquema responde al dibujo de la figura y a continuación veremos su funcionamiento básico.

Su funcionamiento

El microprocesador 6502 funciona de la siguiente manera: contiene un juego de instrucciones codificadas con 8 bits

8 BITS
BUS DATOS
CONTROLADOR BUS DATOS
CONTROL EXTERNO
DECODIF. INSTR.
Reg. instr.
ALU
CONTROL INTERNO
REGISTROS
CONTROLADOR DIR
BUS DIRECCIONES
16 BITS

que entiende y es capaz de ejecutar. Así, por ejemplo, hay una instrucción que consiste en traer un dato de una determinada posición de memoria a uno de los registros internos del 6502. Esta instrucción tendrá un código de 8 bits y cuando se la mandemos al microprocesador, éste lo decodificará y realizará la operación.

Para obtener una instrucción (su código), el microprocesador coloca en el bus de direcciones (16 bits) la dirección de memoria donde se encuentra dicho código y espera a que la memoria le conteste colocándolo en el bus de datos (8 bits). Los controladores de los buses de datos y de direcciones son los encargados de indicar al microprocesador cuando pueden mandar datos o tienen datos preparados para ser leídos.

Una vez que el 6502 tiene el código de la instrucción que debe ejecutar, lo guarda en el registro de instrucciones y el decodificador de instrucciones se encarga de mandar por sus líneas de salida las órdenes necesarias para realizar lo que requiera la instrucción.

Los registros del 6502

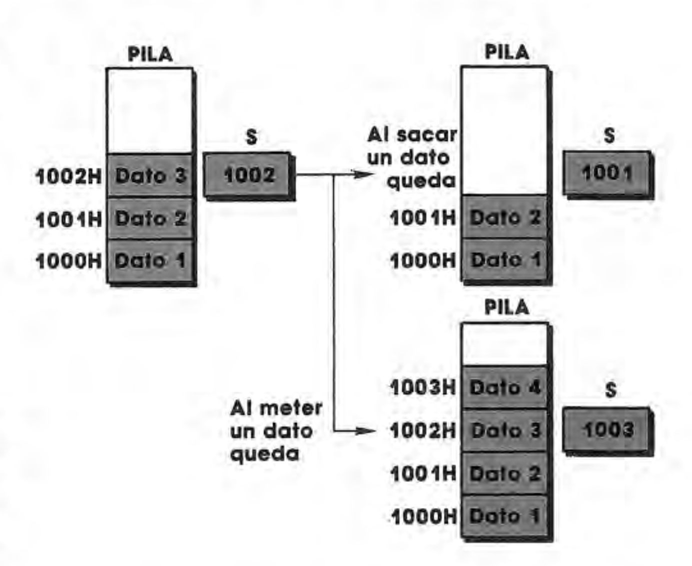
Con lo que hemos visto hasta ahora se nos plantea una pregunta: ¿Cómo sabe el microprocesador en qué dirección de memoria se encuentra la instrucción que debe ejecutar?

El contador de programa (PC), uno de los registros del 6502, nos soluciona este problema. El PC es un registro que contiene la dirección de la instrucción que debe ejecutar el microprocesador en cada momento. Todas las instrucciones que queramos que el 6502 ejecute consecutivamente (un programa) deberán encontrarse en posiciones de memoria consecutivas. De esta forma el PC, cada vez que obtiene un código de instrucción aumenta en uno para contener la dirección del siguiente byte que se debe leer de memoria.

Otros registros importantes del 6502 son:

— **El acumulador.** Podríamos decir que es un registro "comodín" que utiliza el 6502 para realizar gran parte de las operaciones. Así, hay muchas operaciones que sólo pueden realizarse cuando el operando o alguno de ellos se encuentra en el acumulador. Esto nos ahorra tener que especificar en muchas instrucciones los operandos porque ya sabemos que se refiere al número guardado en el acumulador.

— **El puntero de pila (S).** Es un registro que contiene la dirección de memoria de la "cima" de la pila. La pila es un conjunto de direcciones de memoria consecutivas donde guardamos datos con la característica de que el último dato introducido será el primero en sacar. La "cima" de la pila será precisamente la dirección del último dato introducido. Cuando se introduce un dato se aumenta en 1 el puntero de pila y cuando se saca un dato se disminuye en 1.



— **El registro de estado (P).** Es un registro distinto de los demás. No contiene un número codificado con 8 ó 16 bits, como los anteriores, sino que tiene 8 bits cada uno, con un significado distinto. Cada uno de los bits pasará de 0 a 1 cada vez que en el resultado de una operación se produzca alguno de los casos que representa cada bit: resultado negativo, rebosamiento en una operación aritmética, cero, acarreo en una suma, etc. De todos los bits del registro representados en la figura son el 0, 1, 6, 7 los que más nos van a interesar al comenzar a dar nuestros primeros pasos con la programación del 6502.



Los bits del registro de estado nos sirven para realizar instrucciones condicionales según el valor de alguno de los bits, lo que nos indicará lo que haya ocurrido en la última instrucción. Existen en el juego de instrucciones del 6502 algunas que consisten en realizar algo (normalmente un salto a otra dirección de memoria) únicamente cuando se cumpla que alguno de los bits del registro de estado esté a 1



El registro P

También existe el registro índice (I), que veremos a continuación, al estudiar los modos de direccionamiento.

Los bits de cada registro serán, visto lo anterior, los siguientes:

— Acumulador (registro de datos): 8 bits.

— Puntero de pila (registro de direcciones): 16 bits.

— Registro de estado: 8 bits.

— Contador de programa (registro de direcciones): 16 bits.



Las instrucciones del 6502

Como ya hemos dicho anteriormente, las instrucciones del 6502 son códigos de 8 bits. Aquéllas que necesiten algún tipo de operando llevarán a continuación del byte de la instrucción uno o varios bytes con los operandos correspondientes. Así, una instrucción que sea sumar un valor al acumulador, a continuación del byte con el código de instrucción deberá llevar otro byte con el valor que hay que sumar. El PC deberá tener en cuenta este tipo de instrucciones y leer después del código los bytes de datos que contengan. Cuando programamos en lenguaje máquina del 6502 normalmente se escriben los bytes mediante su codificación en hexadecimal (8 bits = 2 cifras hexadecimales).



Métodos de direccionamiento

Las instrucciones del 6502 nos permiten varios sistemas de direccionamiento:

— **Direccionamiento inmediato.** El dato viene en un byte a continuación del byte de instrucción.

— **Direccionamiento absoluto.** El dato está en la dirección de memoria que se introduce como operando en los dos bytes siguientes al de la instrucción.

— **Direccionamiento de página cero.** Es como el absoluto, pero suponiendo que el primer byte de la dirección es 00, por lo que la instrucción únicamente llevará un byte con la segunda parte de la dirección.

— **Direccionamiento indexado.** A la dirección que se introduce en los dos bytes siguientes a la instrucción hay que sumarle el contenido del registro indice (I), de 8 bits, para obtener la dirección donde se encuentra el dato.

— **Direccionamiento indirecto.** En los dos bytes siguientes a la instrucción se encuentra la dirección donde está la dirección donde se encuentra el dato. Esos dos bytes indican dos posiciones de memoria por el procedimiento de página cero.

El 6502 posee instrucciones que permiten obtener cada uno de los datos por los distintos medios de direccionamiento expuestos.



PROGRAMAS

PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

Cambio de coordenadas

Este primer programa, que sirve para todos los ordenadores, es una de las más útiles rutinas para cualquier estudiante o profesional. Por sí misma sirve para poco, pero unida a tus propios programas puede serte de mucha ayuda.

```
CAMBIO DE COORDENADAS
=====================

100 REM ****************************************************************
101 REM *********** CAMBIO DE COORDENADAS DE RECTANGULAR A  **************
102 REM ***********             POLAR Y VICEVERSA           **************
103 REM ****************************************************************
104 REM
105 REM ****************************************************************
106 REM *************** POR JUAN MANUEL GUTIERREZ LEITON  ***************
107 REM ****************************************************************
108 REM
109 REM ****************************************************************
110 REM *************** (c) EDICIONES SIGLO CULTURAL, 1987 **************
111 REM ****************************************************************
112 REM
113 CLS
114 LOCATE 12,20
115 INPUT "de rectangular a polar (S/N)";O$
116 IF O$="s" OR O$="S" THEN GOTO 119
117 IF O$="N" OR O$="n" THEN GOTO 130
118 GOTO 113
119 CLS
120 INPUT "a=";A
121 INPUT "b=";B
122 LET M=SQR(A^2+B^2)
123 IF A=0 THEN IF B<0 THEN LET C=-90:GOTO 129
124 IF A=0 THEN IF B>0 THEN LET C=90: GOTO 129
125 IF A<0 THEN IF B=0 THEN LET C=180:GOTO 129
126 LET C=ATN(B/A)
127 IF A<0 THEN IF B<0 THEN LET C=C-180:GOTO 129
128 IF A<0 THEN IF B>0 THEN LET C=C+180:GOTO  129
129 PRINT M;"[";C;"]":GOTO 138
130 CLS
131 INPUT "modulo=";M
132 INPUT "argumento=";C
```

```
133 LET A=M*COS(C)
134 LET M=M*SIN(C)
135 IF B<0 THEN PRINT A;"-(";ABS(B);")j":GOTO 138
136 IF B>0 THEN PRINT A;"+";B;"j":GOTO 138
137 PRINT A;" + 0j"
138 LOCATE 12,20
139 PRINT "pulse una tecla para continuar"
140 A$=INKEY$:IF A$="" THEN 140
141 CLS
142 LOCATE 1,1
143 PRINT "quiere seguir (S/N)";
144 INPUT A$
145 IF A$="s" OR A$="S" THEN GOTO 100
146 CLS
147 PRINT "A D I O S"
148 PRINT "----------"
149 PRINT
150 PRINT
151 PRINT
152 PRINT
```

El programa puede funcionar perfectamente sin cambios en el IBM pc, xt y at, en todos los compatibles de IBM y en el AMSTRAD. Para el resto de los ordenadores te propongo las siguientes modificaciones.

COMMODORE:

```
113 PRINT CHR$(147)
114 POKE 214,12:POKE 211,19
119 PRINT CHR$(147)
130 PRINT CHR$(147)
138 POKE 214,12:POKE 211,19
140 GET A$:IF A$="" THEN GOTO 140
141 PRINT CHR$(147)
142 POKE 214,1:POKE 211,0
146 PRINT CHR$(147)
```

MSX:

```
114 LOCATE 20,12
138 LOCATE 20,12
142 LOCATE 1,1
```

SPECTRUM:

```
114 PRINT AT 12,2;
138 PRINT AT 12,2;
142 PRINT AT 0,0;
```

La función de este programa es pasar de coordenadas polares a rectangulares y de rectangulares a polares. Al principio del programa (línea 115) se pregunta al usuario cuál de las dos cosas quiere hacer.



Notas sobre el programa 1

En el caso de que utilices esta rutina en uno de tus programas, esta línea la tendrás que quitar, y llamar a la línea 119 para pasar de coordenadas rectangulares a polares, o a la línea 130 para pasar de polares a rectangulares.

Si vas a utilizar este programa como una rutina de tus programas tendrás que hacer las siguientes modificaciones:

1. Quitar el INPUT de la línea 115, así como las líneas 116 y 117, como ya hemos dicho arriba.
2. Quitar las líneas 119, 120 y 121. Los valores de A y B tendrás que darlos por programa.
3. Quitar las líneas 129 y 130. El resultado del programa se te devuelve en las variables numéricas M y C.
4. Quitar las líneas 131 y 132. Si eliges pasar de coordenadas polares a rectangulares, los valores de M y de C tendrás que darlos por programa.
5. Quitar las líneas que van desde la 135 hasta el final del programa. El resultado de la operación se devuelve en A y en B.
6. Tendrás que poner en alguna parte del programa un RETURN para que, cuando termine el cálculo, se devuelva el control al programa principal.



Awari

Este juego de mesa (o de suelo) es originario del antiguo Egipto. También se le conoce con el nombre de WARI o MANCANA. Los africanos dicen que es un «JUEGO DE HOMBRES», pero también pueden jugar a él las mujeres, y de hecho, cuando no están sus maridos, juegan.

La versión que proponemos aquí es un poco más sencilla que la que se juega en la realidad. Esto es así para no hacer el programa más largo de lo que ya es y para que las reglas del juego no sean tan complicadas.

```
AWARI
=====

10 REM        **************************************************
11 REM        **                                              **
12 REM        **       AAAA  W     W  AAAA  RRRRR  IIIII      **
13 REM        **      A    A W     W A    A R    R   I       **
14 REM        **      A    A  W W W  A    A R    R   I       **
15 REM        **      AAAAAA  W W W  AAAAAA RRRRR    I       **
16 REM        **      A    A   W W   A    A R R      I       **
17 REM        **      A    A   W W   A    A R  R   IIIII     **
18 REM        **                                              **
19 REM        **                                              **
20 REM        **************************************************
22 REM
23 REM **************************************************************
24 REM **************************************************************
25 REM *****      REALIZADO POR FRANCISCO MORALES GUERRERO  *********
26 REM **************************************************************
27 REM
28 REM **************************************************************
29 REM **************************************************************
30 REM *****     (c) EDICIONES SIGLO CULTURAL, 1987        *********
31 REM **************************************************************
32 REM
100 LET N=0
101 DIM B(13):DIM G(13):DIM F(50)
102 CLS
103 PRINT "****************************************"
104 FOR I=1 TO 17
105    PRINT "*";TAB(40);"*"
106 NEXT I
107 PRINT "****************************************"
108 GOSUB 306
109 LOCATE 16,3:PRINT "(c) Ed. Siglo Cultural"
110 FOR I=1 TO 2000
111 NEXT I
112 LOCATE 12,3
113 PRINT "Quieres ver las instrucciones S/N "
114 LET A$=INKEY$:IF A$="" THEN GOTO 114
115 IF A$="n" OR A$="N" THEN GOTO 118
116 IF A$<>"s" AND A$<>"S" THEN GOTO 112
117 GOSUB 277
118 CLS
119 GOSUB 306
120 LOCATE 24,1
121 FOR I=1 TO 4
122    PRINT
123 NEXT I
124 LOCATE 6,1:PRINT "========================================"
125 PRINT
126 PRINT
127 LET E=0
128 FOR I=0 TO 12
129    LET B(I)=3
130 NEXT I
131 LET C=0
132 LET F(N)=0
133 LET B(13)=0
134 LET B(6)=0
135 GOSUB 206
136 LOCATE 16,1
137 PRINT "
138 LOCATE 16,1
139 PRINT "MUEVES TU ";:GOSUB 174
140 IF E=0 THEN GOTO 159
141 IF M=H THEN GOSUB 170
```

```
142 IF E=0 THEN GOTO 159
143 LOCATE 16,1
144 PRINT "                                         "
145 LOCATE 16,1
146 PRINT "MI MOVIMIENTO ES ";
147 GOSUB 240
148 IF E=0 THEN GOTO 159
149 IF M=H THEN PRINT ",";:GOSUB 240
150 FOR I=1 TO 1500
151 NEXT I
152 IF E>0 THEN GOTO 135
153 LOCATE 16,1
154 PRINT "                                         "
155 LOCATE 16,1
156 PRINT "PULSA UNA TECLA PARA VER MI MOVIMIENTO"
157 LET A$=INKEY$:IF A$="" THEN GOTO 157
158 IF E>0 THEN GOTO 135
159 PRINT
160 FOR I=1 TO 1500
161 NEXT I
162 LOCATE 16,13
163 PRINT "                                         "
164 PRINT "FIN DEL JUEGO.":PRINT
165 LET D=B(6)-B(13)
166 IF D<0 THEN PRINT "TE HE GANADO POR ";-D;" PUNTOS":GOTO 315
167 LET N=N+1
168 IF D=0 THEN PRINT "HEMOS EMPATADO":GOTO 315
169 PRINT "ME HAS GANADO POR ";D;"PUNTOS":GOTO 315
170 LOCATE 16,1
171 PRINT "                                         "
172 LOCATE 16,1
173 PRINT "MUEVE OTRA VEZ ";
174 INPUT M
175 IF M<7 THEN IF M>0 THEN LET M=M-1:GOTO 188
176 LOCATE 16,1
177 PRINT "                                         "
178 LOCATE 16,1
179 PRINT "                                         "
180 LOCATE 16,1
181 PRINT "MOVIMIENTO NO PERMITIDO"
182 BEEP:BEEP
183 FOR I=1 TO 1000
184 NEXT I
185 LOCATE 16,1
186 PRINT "                                         "
187 GOTO 170
188 IF B(M)=0 THEN GOTO 179
189 LET H=6
190 GOSUB 192
191 GOTO 206
192 LET K=M
193 GOSUB 227
194 LET E=0
195 IF K>6 THEN LET K=K+7
196 LET C=C+1
197 IF C<9 THEN LET F(N)=F(N)*6+K
198 FOR I=0 TO 5
199    IF B(I)<>0 THEN GOTO 202
200 NEXT I
201 RETURN
202 FOR I=7 TO 12
203    IF B(I)<>0 THEN LET E=1:RETURN
204 NEXT I
205 RETURN
206 LOCATE 10,1
207 PRINT
208 PRINT "      ";
209 FOR I=12 TO 7 STEP -1
210    GOSUB 224
211 NEXT I
212 PRINT:PRINT "    ";
213 LET I=13
214 GOSUB 224
215 PRINT "                         ";
216 PRINT B(6)
217 PRINT "       ";
218 FOR I=0 TO 5
219    GOSUB 224
220 NEXT I
221 PRINT
222 PRINT
223 RETURN
```

```
224 IF B(I)<10 THEN PRINT " ";
225 PRINT B(I);
226 RETURN
227 LET P=B(M)
228 LET B(M)=0
229 FOR P=P TO 1 STEP -1
230     LET M=M+1
231     IF M>13 THEN LET M=M-14
232     LET B(M)=B(M)+1
233 NEXT P
234 IF B(M)=1 THEN IF M<>6 THEN IF M<>13 THEN IF B(12-M)<>0 THEN GOTO 236
235 RETURN
236 LET B(H)=B(H)+B(12-M)+1
237 LET B(M)=0
238 LET B(12-M)=0
239 RETURN
240 LET D=-99
241 LET H=13
242 FOR I=0 TO 13
243     LET G(I)=B(I)
244 NEXT I
245 FOR J=7 TO 12
246     IF B(J)=0 THEN GOTO 269
247     LET G=0
248     LET M=J
249     GOSUB 227
250     FOR I=0 TO 5
251         IF B(I)=0 THEN GOTO 257
252         LET L=B(I)+I
253         LET R=0
254         IF L>13 THEN LET L=L-14:LET R=1:GOTO 254
255         IF B(L)=0 THEN IF L<>6 THEN IF L<>13 THEN LET R=B(12-L)+R
256         IF R>Q THEN LET Q=R
257     NEXT I
258     LET Q=B(13)-B(6)-Q
259     IF C>8 THEN GOTO 265
260     LET K=J
261     IF K>6 THEN LET K=K-7
262     FOR I=0 TO N-1
263         IF F(N)*6+K=INT(F(I)/6^(7-C)+.1) THEN LET Q=Q-2
264     NEXT I
265     FOR I=0 TO 13
266         LET B(I)=G(I)
267     NEXT I
268     IF Q>=D THEN LET A=J:LET D=Q
269 NEXT J
270 LET M=A
271 PRINT CHR$(42+M);
272 GOTO 192
273 FOR I=0 TO N-1
274     PRINT B(I)
275 NEXT I
276 END
277 REM
278 REM *****************
279 REM * INSTRUCCIONES *
280 REM *****************
281 REM
282 CLS
283 PRINT TAB(14);"INSTRUCCIONES"
284 PRINT TAB(13);"---------------"
285 PRINT
286 PRINT " El juego del AWARI (tambien llamado    WARI o MANCALA) es muy antigu
o, tanto   que ya se practicaba en el antiguo      egipto."
287 PRINT:PRINT " La forma de jugar es la siguiente:"
288 PRINT
289 PRINT " El tablero esta dividido en seis casi- llas en cada cara. Los movimi
entos se    realizan cogiendo todas las fichas de"
290 PRINT "una cierta casilla que, no este vacia, yrepartiendo dichas fichas por
 las casi- llas contiguas hacia la derecha, ponien-"
291 PRINT "do una en cada casilla."
292 PRINT
293 PRINT " Cada torno puede ser de una o dos juga-das. Se realizara una segunda
 jugada    cuando ninguna de las fichas pasa al la-do del oponente."
294 PRINT:PRINT "PULSA UNA TECLA";
295 LET A$=INKEY$:IF A$="" THEN GOTO 295
296 CLS
297 PRINT " Para ganar hay que conseguir apropiarsede las fichas del oponente. E
sto se con-sigue cuando despues de un movimiento,  la ultima ficha cae en un cas
illero que"
298 PRINT "tiene solo una, dos o tres fichas. Todaslas fichas de dicho cubilete
son comidas"
299 PRINT
```

```
300 PRINT " El juego termina cuando una de las ca- ras del tablero esta vacia."
301 PRINT
302 PRINT "PULSA UNA TECLA"
303 LET A$=INKEY$:IF A$="" THEN GOTO 303
304 CLS
305 RETURN
306 REM ************************
307 REM * ROTULO     "A W A R I" *
308 REM ************************
309 REM
310 LOCATE 5,5:PRINT " ###  #     #  ###  ####  #####"
311 LOCATE 6,5:PRINT "#   #  #   #  #   # #   #   #"
312 LOCATE 7,5:PRINT "#####  # # #  ##### ####    #"
313 LOCATE 8,5:PRINT "#   #   # #   #   # #  #  #####"
314 RETURN
315 PRINT:PRINT
316 PRINT "(HECHAMOS OTRA PARTIDA? (S/N) ";
317 LET A$=INKEY$:IF A$="" THEN GOTO 317
318 IF A$="s" OR A$="S" THEN GOTO 118
319 IF A$<>"n" AND A$<>"N" THEN GOTO 317
320 CLS
321 GOSUB 306
322 LOCATE 24,1
323 FOR I=1 TO 4
324    PRINT
325 NEXT I
326 LOCATE 10,1
327 PRINT
328 PRINT "  ****   *****   *****   ****    ****"
329 PRINT " *    *  *    *    *    *    *  *    *"
330 PRINT " *    *  *    *    *    *    *  *"
331 PRINT " ******  *    *    *    *    *   ****"
332 PRINT " *    *  *    *    *    *    *       *"
333 PRINT " *    *  *    *    *    *    *  *    *"
334 PRINT " *    *  *****   *****   ****    ****"
335 PRINT
336 PRINT
337 END
```

Las reglas del juego son las siguientes:

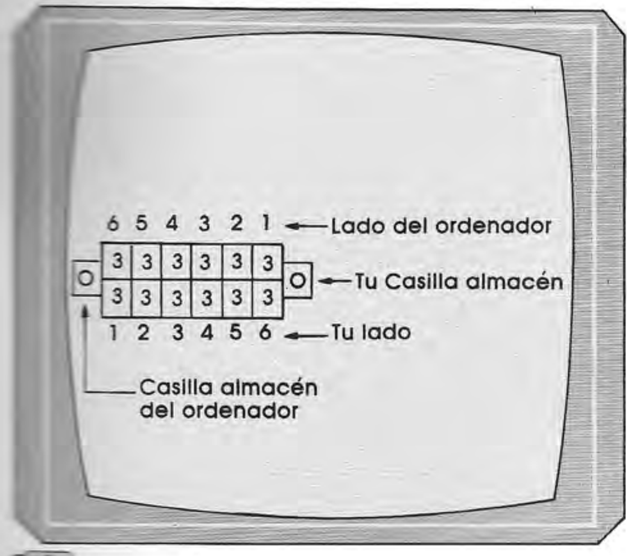
Este juego se realiza en una especie de tablero como el que se ve en la figura 1. En realidad cada casilla, aunque la llamaremos así, no es tal casilla, sino que es un cuenco o un recipiente medianamente hondo dentro del cual pondremos las fichas.

En la versión para el ordenador hemos decidido poner tres fichas en cada recipiente.

Las dos casillas que están vacías son las que almacenarán las fichas que le vayamos comiendo al enemigo. Este es el objetivo primordial del juego. Pero el juego no termina cuando nos hemos comido todas las fichas del contrincante, pues a veces es imposible saber cuál son las propias y cuáles las del contrincante, sino que lo hace cuando una de las caras se ha quedado sin fichas. En ese momento se ve la diferencia de fichas que hay entre los dos jugadores y gana el que más tiene.

Para mover las fichas sólo hay que decir el número de casilla sobre la que queremos actuar. Una vez sabido esto, se reparten todas las fichas que haya en dicha casilla entre las casillas contiguas por la derecha y poniendo una ficha en cada casilla.

Si nos fijamos en la figura 1, veremos que en la posición a, en la casilla 3 hay tres fichas. Si movemos desde la casilla 3, ésta se quedará vacía y se pondrá una ficha en la 4, otra en la 5 y otra en la 6. Con ello estas tres casillas pasarán a tener 4 fichas.

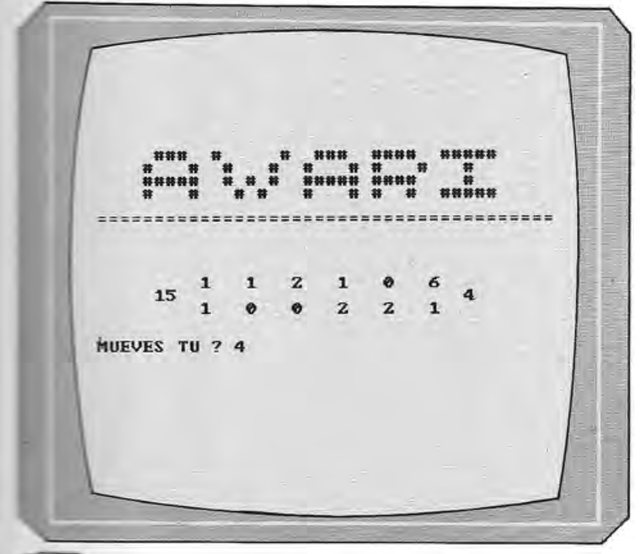




Tablero de Awari.

Se come una ficha del contrincante cuando, al hacer un movimiento, se cambia de lado (se pasa al lado del contrincante) y la última de las fichas a colocar cae en una casilla que no esté vacía y que tenga como máximo tres fichas. El jugador que haga dicha jugada cogerá todas las fichas de dicho cuenco y las pondrá en el cuenco de recuento.

Cabe la posibilidad de hacer dos jugadas seguidas. Esto sucede cuando la última ficha ha caído en el primer casillero del contrincante.





Ejemplo de ejecución del programa Awari.

Este programa funciona en el IBM pc, xt, at y compatibles, así como en el AMSTRAD. Para los demás ordenadores os damos las variaciones:

COMMODORE:

```
102 PRINT CHR$(147)
109 POKE 214,16:POKE 211,2:PRINT "(c)
Ed. Siglo Cultural
112 POKE 214,12:POKE 211,2
114 GET A$:IF A$="" THEN GOTO 114
120 POKE 214,23:POKE 211,0
124 POKE 214,6:POKE 211,0:PRINT
"================"
136 POKE 214,16:POKE 211,0
138 POKE 214,16:POKE 211,0
143 POKE 214,16:POKE 211,0
145 POKE 214,16:POKE 211,0
153 POKE 214,16:POKE 211,0
155 POKE 214,16.POKE 211,0
157 GET A$:IF A$="" THEN GOTO 157
170 POKE 214,16:POKE 211,0
172 POKE 214,16:POKE 211,0
176 POKE 214,16:POKE 211,0
178 POKE 214,16:POKE 211,0
180 POKE 214,16:POKE 211,0
185 POKE 214,16.POKE 211,0
206 POKE 214,10:POKE 211,0
282 PRINT CHR$(147)
295 GET A$:IF A$="" THEN GOTO 295
296 PRINT CHR$(147)
303 GET A$:IF A$="" THEN GOTO 303
304 PRINT CHR$(147)
310 POKE 214,5:POKE 211,5
311 POKE 214,6:POKE 211,5
312 POKE 214,7:POKE 211,5
313 POKE 214,8:POKE 211,5
320 PRINT CHR$(147)
326 POKE 214,10:POKE 211,0
```

MSX:

Lo único que hay que hacer para que el programa funcione en el MSX es cambiar el orden de los argumentos de todas las sentencias LOCATE. Esto es, si aparece la instrucción LOCATE 12,1 nosotros tendremos que poner LOCATE 1,12.

La versión para SPECTRUM de este programa aparecerá en tomos posteriores.



Letras en tres dimensiones para SPECTRUM

Este programa es una rutina en CODIGO MAQUINA para el SPECTRUM que nos permitirá la impresión en pantalla de cualquier mensaje con las letras en tres dimensiones.

La rutina no es reubicable. Esto es, no se puede localizar en cualquier parte de

la memoria, sino sólo en la posición 64500 de ésta. No es reubicable debido a que tiene saltos absolutos y subrutinas.

```
  1 POKE 23658,0: CLEAR 64499:
GO SUB 1000
  2 CLS : PRINT AT 10,1; INK 6;
"QUIERES GRABAR EL PROGRAMA S/N"
  3 IF INKEY$="n" THEN  GO TO 6
  4 IF INKEY$="s" THEN  PRINT A
T 15,9; FLASH 1;" PULSA ENTER ":
 PAUSE 10: FOR I=1 TO 50: NEXT I
: PAUSE 0: POKE 23736,181: SAVE
"LETRAS 3-D" LINE 1: GO TO 6
  5 GO TO 3
  6 CLS : PAUSE 3: FOR i=0 TO 3
0: NEXT i: PRINT AT 10,1; INK 5;
" GRABO EL CODIGO MAQUINA  S/N "
  7 IF INKEY$="n" THEN  GO TO 9
990
  8 IF INKEY$="s" THEN  PRINT A
T 15,9; FLASH 1;" PULSA ENTER ":
 PAUSE 3: FOR I=1 TO 50: NEXT I:
 PAUSE 0: POKE 23736,181: SAVE "
L. 3-DCODE "CODE 64500,483: GO T
O 9990
  9 GO TO 7

 10 REM
POSICION         SIGNIFICADO
                LONGITUD DEL MEN
  23300         SAJE MULTIPLICADO
                POR OCHO.


 11 REM
POSICION         SIGNIFICADO
 23302             XSIZE

 23303             YSIZE

 23311             XPOS

 23312             YPOS

 15 PRINT  INK 0;AT 20,0;M$: PO
KE 23300,LEN M$*8
 20 POKE 23311,XPOS: POKE 23312
,YPOS
 30 POKE 23302,XSIZE: POKE 2330
3,YSIZE
 40 INK I:

        RANDOMIZE USR 64500

 50 RETURN
1000 INK 6: PAPER 0: BORDER 0: C
LS : PRINT AT 0,6; INK 4; 1985
, Fco. Morales.": INK 5: PLOT 40
,165: DRAW 179,0
1010 INK 6: PRINT ''" ESTE ES UN
 PROGRAMA QUE TE PER-MITIRA LA C
REACION  DE  MENSAJESDE CUALQUIE
R TIPO EN TRES DIMEN-SIONES."
1020 PRINT ''" PARA UTILIZARLO S
OLO HAS DE DAREL MENSAJE A IMPRI
MIR, LA  POSI-CION DONDE QUIERES
 QUE LO IMPRI-MA  Y LA ALTURA Y
ANCHURA  EN LAQUE LO QUIERES."
1030 INK 7: PRINT ''"- EL MENSAJ
E IRA EN M$"''"- LA POSICION EN
XPOS E YPOS"''"- LA ALTURA Y ANC
HURA EN YSIZE Y XSIZE RESPECTIVA
MENTE."
1040 PRINT #0; INK 2; FLASH 1;"
PULSA UNA TECLA PARA CONTINUAR "
1050 PAUSE 0
1060 INK 6: CLS : PRINT AT 9,9;
FLASH 1;"CARGANDO DATAS";AT 11,6
; FLASH 1; INVERSE 1; INK 4;" ES
PERA UN MOMENTO. "
1070 PRINT AT 15,6; INK 5;"CHECK
SUM = "
1080 LET CONTA=0: LET POS=64500:
 RESTORE 7000
1090 FOR I=0 TO 482: READ A: POK
E POS+I,A: LET CONTA=CONTA+A: PR
INT AT 15,17;CONTA;: NEXT I
1092 IF CONTA<>40670 THEN  GO TO
 6000
1100 CLS : PRINT AT 10,0; INK 4;
"LETRAS 3-D INSTALADO EN LA 6450
0"
1110 PRINT AT 12,0; FLASH 1;"PUL
SA UNA TECLA PARA DEMOSTRAION":
PAUSE 0
1120 LET M$="letras": REM MENSAJ
E
1130 LET XPOS=0: REM POSICION EN
 X
1140 LET YPOS=95: REM POSICION E
N Y
1150 LET XSIZE=4: REM ANCHURA
1160 LET YSIZE=11: REM ALTURA
1170 LET I=2: CLS : GO SUB 10
1180 LET I=4: LET M$="3-D   ": L
ET XPOS=39: LET YPOS=40: LET XSI
ZE=9: LET YSIZE=6: GO SUB 10
1190 LET I=5: LET M$="PULSA UNA
TECLA": LET XPOS=6: LET YPOS=16:
 LET XSIZE=2: LET YSIZE=2: GO SU
B 10
1200 PAUSE 0: RETURN
6000 CLS : PRINT  INK 4;AT 10,1;
 FLASH 1;"ERROR"; FLASH 0;" EN D
ATAS,PRUEBA DE NUEVO"; INK 6;AT
15,9; FLASH 1;" PULSA ENTER ": P
AUSE 3: PAUSE 0: GO TO 20000
6500
6510 REM *********************
6520 REM ***** D A T A S *****
6530 REM *********************
6540

7000 DATA 0,217,229,217,58,4,91,
50,17,91,50,19,91,62,15,50,18,91
,50,5,91,58,6,91,71

7005 DATA 5,58,17,91,79,167,129,
16,252,71,58,15,91,167,128,50,10
,91,58,7,91,71,5,58,18

7010 DATA 91,214,8,79,167,129,16
,252,71,58,16,91,167,128,50,11,9
1,205,101,252,58,18,91,61,254

7015 DATA 7,50,18,91,50,5,91,32,
193,62,15,50,18,91,50,5,91,58,17
,91,60,42,19,91,189

7020 DATA 50,17,91,50,4,91,32,16
9,217,225,217,251,201,62,0,50,14
,91,205,169,253,42,4,91 68

7025 DATA 77,205,160,252,202,168
,253,58,6,91,71,197,58,10,91,128
,79,58,11,91,71,205,229,34,14

7030  DATA 0,58,7,91,71,17,255,2
55,205,186,36,193,120,61,71,254,
255,32,223,195,172,252,205,170,3
4
7035 DATA 71,4,126,7,16,253,203,
71,201,58,4,91,79,58,5,91,60,71,
205,160,252,32,39,58,4

7040 DATA 91,61,79,58,5,91,71,20
5,160,252,32,25,58,10,91,79,58,1
1,91,71,205,229,34,58,6
```

```
7045 DATA 91,79,58,7,91,71,17,1,
1,205,186,36,58,4,91,79,58,5,91,
61,71,205,160,252,32

7050 DATA 39,58,4,91,60,79,58,5,
91,71,205,160,252,32,25,58,8,91,
79,58,13,91,71,205,229

7055 DATA 34,58,6,91,79,58,7,91,
71,17,1,1,205,186,36,58,4,91,79,
58,5,91,60,71,205

7060 DATA 160,252,32,23,58,8,91,
79,58,9,91,71,205,229,34,58,6,91
,79,6,0,17,1,1,205

7065 DATA 186,36,58,4,91,60,79,5
8,5,91,71,205,160,252,32,23,58,1
2,91,79,58,9,91,71,205

7070 DATA 229,34,14,0,58,7,91,71
,17,255,255,205,186,36,58,4,91,7
9,58,5,91,60,71,205,160

7075 DATA 252,40,5,62,1,50,14,91
,58,4,91,60,79,58,5,91,71,205,16
0,252,40,7,58,14,91

7080 DATA 60,50,14,91,58,14,91,2
54,1,40,25,58,8,91,79,58,11,91,7
1,205,229,34,58,6,91

7085 DATA 79,58,7,91,71,17,1,1,2
05,186,36,201,58,6,91,71,58,10,9
1-128,50,8,91,58,7

7090 DATA 91,71,58,11,91,128,50,
9,91,58,6,91,135,71,58,10,91,128
,50,12,91,58,7,91,71

7095 DATA 58,11,91,144,50,13,91,
201,0

9990 CLS : PRINT AT 0,6; INK 4;"
 1985, Fco. Morales.";©0;AT 1,9
; INVERSE 1;" LETRAS 3-D ": INK
5: PLOT 40,165: DRAW 179,0
9999 INK 3: FOR i=0 TO 3: PLOT 1
0+i,20+i: DRAW 0,110-2*i: DRAW 2
35-2*i,0: DRAW 0,-110+2*i: DRAW
-235+2*i,0: NEXT i: LET i=6: LET
 m$="ADIOS": LET XPOS=25: LET YP
OS=40: LET XSIZE=5: LET YSIZE=10
: GO SUB 10: PAUSE 0
```

Lo primero que tenemos que hacer es introducir el programa por el teclado del ordenador. Ten mucho cuidado con los números de las líneas DATA para no equivocarte. De todas maneras el programa lleva una rutina de comprobación que te dirá si te has equivocado.

Una vez que esté el programa en memoria ejecútalo con la sentencia RUN y espera que el ordenador meta el código máquina en su memoria. Una vez hecho esto te aparecerá en pantalla una demostración de lo que es capaz de hacer. Cuando el programa te pregunte si quieres grabarlo, introduce en tu cassette una cinta virgen rebobinada, pulsa a la vez las teclas RECORD y PLAY, espera unos 10 segundos y pulsa la tecla S del SPECTRUM. A continuación se grabará el programa en la cinta.





Demostración del programa «Letras 3-D».

Una vez que se haya grabado el programa el SPECTRUM te preguntará si quieres grabar la rutina en CODIGO MAQUINA. Si te interesa hacerlo, sólo tienes que pulsar la tecla S y el ordenador lo hará.



Notas sobre el programa 3

Para utilizar esta rutina en tus programas sólo tienes que hacer lo siguiente:

1. Introducir esta línea al principio del programa:

```
10 CLEAR 64499:LOAD ""CODE
```

2. Introducir esta rutina al final del programa:

```
9950 PRINT INK 0; AT 20,0;M$:POKE 23300,LEN M$*8
9960 POKE 23311,XPOS:POKE 23312,YPOS
9970 POKE 23302,XSIZE:POKE 23303,YSIZE
9980 INK I : RANDOMIZE USR 64500
9990 RETURN
```

Para usar esta rutina debes de saber que:

— El mensaje a imprimir ha de ir almacenado en M$.

— La posición en la pantalla, donde aparecerá el mensaje, ha de ir en XPOS

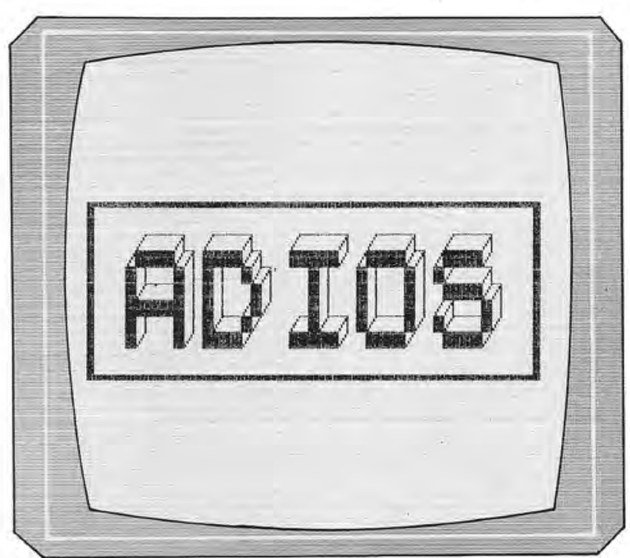
y en YPOS. Siendo XPOS la coordenada X de la pantalla y YPOS la coordenada Y.

— La altura del mensaje tiene que ir almacenado en YSIZE.

— La anchura debe ir en XSIZE.

— El color tiene que ir en I.

— Si necesitas efectos de FLASH o de BRIGHT, tendrás que ponerlos como una instrucción normal antes de llamar a la rutina.





Mensaje de despedida del programa 1.

Este programa empieza en la dirección 64500 y llega hasta el final de la memoria. Internamente utiliza la zona intermedia de impresora para almacenar una serie de variables. Las más importantes son:

— 23300. En esta dirección ha de ir la longitud de la cadena multiplicada por ocho.

— 23302. XSIZE

— 23303. YSIZE

— 23311. XPOS

— 23312. YPOS

El mensaje a imprimir ha de estar impreso en la columna 0 de la línea 20.

Para llamar a la rutina sólo hay que hacer:

RANDOMIZE USR 64500



Supergráficos

Este es un programa realmente útil para todo programador o usuario. Nos servirá para dibujar en la pantalla de nuestro ordenador cualquier tipo de dibujo en color o en blanco y negro. También nos permitirá copiar el contenido de la pantalla en la impresora.

Este programa sólo funciona en los ordenadores IBM pc, xt, at y compatibles. Para el resto de los ordenadores irá apareciendo en sucesivos tomos.

```
SUPERGRAFICOS
-------------

10 REM *************************************************************************
12 REM *                                                                       *
13 REM *  SSS  U   U PPPP  EEEEE RRRR   GGG  RRRR   AAA  FFFFF III  CCC   SSS  *
14 REM * S   S U   U P   P E   E R   R G   G R   R A   A F   F  I  C   C S   S *
15 REM * S     U   U P   P E E   R   R G     R   R A   A F F    I  C     S     *
16 REM *  SSS  U   U PPPP  EEE   RRRR  G  GG RRRR  AAAAA FFF    I  C      SSS  *
17 REM *     S U   U P     E E   RR    G   G RR    A   A F F    I  C         S *
18 REM * S   S U   U P     E   E R R   G   G R R   A   A F      I  C   C S   S *
19 REM *  SSS   UUU  P     EEEEE R  R   GGG  R  R  A   A F     III  CCC   SSS  *
20 REM *                                                                       *
22 REM *************************************************************************
23 REM
25 REM *************************************************************************
26 REM **************** REALIZADO POR FRANCISCO MORALES GUERRERO ***************
27 REM *************************************************************************
28 REM
30 REM *************************************************************************
31 REM ******************* (c) EDICIONES SIGLO CULTURAL, 1987 ******************
32 REM *************************************************************************
33 REM
36 REM * INICIALIZACION DEL PROGRAMA *
38 REM
39 KEY OFF
40 FOR RT=1 TO 10
41    KEY(RT) OFF
42 NEXT RT
43 SCREEN 1
```

```
44 COLOR 16,1,,,3
45 WIDTH 80
46 LET X=100
47 LET Y=100
48 LET X1=0
49 LET Y1=0
50 LET SS=1
51 DIM A%(10000),B%(10000),C%(3000),D%(2000),E%(2000)
55 GOSUB 173
56 CLS
60 LOCATE 21,1
61 INPUT "Numero de puntos en X (320 o 640) = ";A
62 IF A<>320 AND A<>640 THEN 60 ELSE IF A=320 THEN WIDTH 40:LET SW=1 ELSE LET SW
=2
63 REM
64 REM **********************
65 REM * PROGRAMA PRINCIPAL *
66 REM **********************
67 REM
68 LET PP=0
69 LET II=1
70 COLOR PP,II
71 CLS
72 GOSUB 105
73 GET (0,0)-(319*SW,199),B%
74 P=POINT(X,Y):IF P<>0 THEN P1=0 ELSE P1=1
75 PSET(X,Y),P1
76 A$=INKEY$:IF A$="" THEN 76
77 A=ASC(A$)
78 IF LEN(A$)=1 THEN 83
79 IF MID$(A$,2,1)="K" THEN PSET(X,Y),P:X=X-SS-SS*(X-SS<0):GOTO 74
80 IF MID$(A$,2,1)="M" THEN PSET(X,Y),P:X=X+SS+SS*(X+SS>319*SW):GOTO 74
81 IF MID$(A$,2,1)="H" THEN PSET(X,Y),P:Y=Y-SS-SS*(Y-SS<0):GOTO 74
82 IF MID$(A$,2,1)="P" THEN PSET(X,Y),P:Y=Y+SS+SS*(Y+SS>191):GOTO 74
83 IF A=83 OR A=115 THEN M$="ESCALA ACTUAL ="+STR$(SS)+"                    I
NTRODUZCA NUEVA ESCALA (1-9)":LO=2:MA$="9":MI$="1":GOSUB 129:IF VAL(D$)>9 AND D$
<>"" THEN 83 ELSE SS=VAL(D$)
84 IF A=80 OR A=112 THEN PSET(X,Y),II:X1=X:Y1=Y:SWW=1:GOTO 74
85 IF A=79 OR A=111 THEN PSET(X,Y),PP:X1=X:Y1=Y:SWW=2:GOTO 74
86 IF A=67 OR A=99 THEN LO=2:MA$="4":MI$="0":M$="COLOR":GOSUB 129:IF VAL(D$)>4 A
ND D$<>"" THEN 85 ELSE II=VAL(D$)
87 IF A=76 OR A=108 THEN PSET(X,Y),P:GOSUB 105:LINE(X1,Y1)-(X,Y),-II*(SWW=1)-PP*
(SWW=2):X1=X:Y1=Y:GOTO 74
88 IF A=70 OR A=102 THEN PSET(X,Y),P:GOSUB 105:DD=POINT(X,Y):FOR I=X TO 319*SW:I
F POINT(I,Y)<>DD THEN DD=POINT(I,Y):PAINT(X,Y),II,DD:GOTO 74 ELSE NEXT I:GOTO 74
89 IF A=27 THEN PSET(X,Y),P:GOSUB 111:GOTO 74
90 IF A=75 OR A=107 THEN PSET(X,Y),P:GET (0,0)-(319,191),B%:GOTO 74
91 IF A=74 OR A=106 THEN PSET(X,Y),P:PUT (0,0),B%,PSET:GOTO 74
92 IF A=81 OR A=113 THEN PSET(X,Y),P:M$="( TERMINAMOS EL PROGRAMA ? (S/n)":MA$="
z":MI$="A":LO=2:GOSUB 129:IF D$<>"n" AND D$<>"N" AND D$<>"S" THEN 92 ELSE IF D$=
"S" THEN 152
93 IF A=90 OR A=122 THEN PSET(X,Y),P:M$="( BORRAMOS LA PANTALLA ? (S/n)":MI$="A"
:MA$="z":LO=2:GOSUB 129:IF D$<>"N" AND D$<>"n" AND D$<>"S" THEN 93 ELSE IF D$="S
" THEN CLS:GOTO 74
94 IF A=68 OR A=100 THEN PSET(X,Y),P:GOSUB 105:N$="DIRECTORIO":GOSUB 166:GOSUB 1
58:PRINT "PULSA UNA TECLA":A$=INPUT$(1):GOSUB 111:GOTO 74
95 IF A=66 OR A=98 THEN PSET(X,Y),P:GOSUB 105:N$="BORRAR UN PROGRAMA":GOSUB 166:
GOSUB 158:INPUT "NOMBRE DEL FICHERO = ",A$:KILL A$:GOSUB 111:GOTO 74
96 IF A=82 OR A=114 THEN PSET(X,Y),P:GOSUB 105:N$="RENOMBRAR UN PROGRAMA":GOSUB
166:GOSUB 158:INPUT "NOMBRE ANTIGUO DEL FICHERO = ",A$:PRINT:INPUT"NOMBRE MODERN
O DEL FICHERO = ",B$:NAME A$ AS B$:GOSUB 111:GOTO 74
97 IF A=77 OR A=109 THEN PSET(X,Y),P:DEF SEG=&HB800:M$="NOMBRE PARA GRABAR = ":M
A$="z":MI$="A":LO=12:GOSUB 129:IF D$="" THEN GOTO 74 ELSE BSAVE D$,0,16300:GOTO
74
98 IF A=78 OR A=110 THEN PSET(X,Y),P:DEF SEG=&HB800:M$="NOMBRE PARA GARGAR = ":M
A$="z":MI$="A":LO=12:GOSUB 129:IF D$="" THEN GOTO 74 ELSE BLOAD D$:GOTO 74
99 IF A=88 OR A=120 THEN PSET(X,Y),P:GOSUB 117:LOCATE 22,1:PRINT "X=";X;"  ";"Y=
";Y:LOCATE 23,1:PRINT "PULSA UNA TECLA:"A$=INPUT$(1):GOSUB 123
100 IF A=63 THEN PSET(X,Y),P:GOSUB 173
101 IF (A=87 OR A=119) AND SSW=0 THEN PSET(X,Y),P:X2=X:Y2=Y:SSW=1:GOTO 74
102 IF (A=87 OR A=119) AND SSW=1 THEN PSET(X,Y),P:GET (X2,Y2)-(X,Y),D%:AX=X-X2:A
Y=Y-Y2:X=X2:Y=Y2:GET (X,Y)-(X+AX,Y+AY),E%:SSW=0:GOTO 204
103 PSET(X,Y),P:GOTO 74
104 REM
105 REM *********************
106 REM * ALMACENA PANTALLA *
107 REM *********************
108 REM
109 GET (0,0)-(319*SW,199),A%:RETURN
110 REM
```

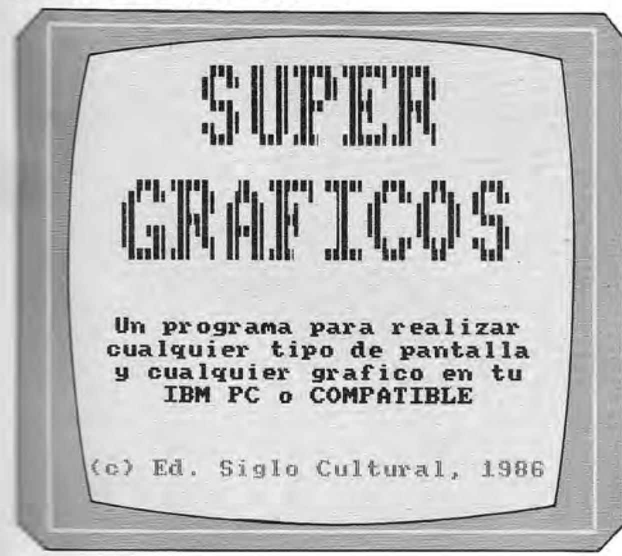
```
111 REM *********************
112 REM * RECUPERA PANTALLA *
113 REM *********************
114 REM
115 PUT (0,0),A%,PSET:RETURN
116 REM
117 REM ********************************
118 REM * ALMACENA 3 LINEAS INFERIORES *
119 REM ********************************
120 REM
121 GET (0,160)-(319*SW,199),C%:RETURN
122 REM
123 REM ********************************
124 REM * RECUPERA 3 LINEAS INFERIORES *
125 REM ********************************
126 REM
127 PUT (0,160),C%,PSET:RETURN
128 REM
129 REM ********************
130 REM * ENTRADA DE DATOS *
131 REM ********************
132 REM
133 GOSUB 117
134 LOCATE 21,1
135 PRINT SPC(120*SW);
136 LET D$=""
137 LET LO=0
138 LOCATE 22,1
139 PRINT M$;" : ";CHR$(29);
140 C$=INKEY$
141 IF C$="" THEN GOTO 140
142 IF C$=CHR$(8) AND LO>0 THEN LO=LO-1:D$=LEFT$(D$,LO):PRINT CHR$(29);"_ ";CHR$
(29);CHR$(29);
143 IF C$=CHR$(13) THEN GOTO 150
144 IF C$=" " OR C$="." THEN GOTO 146
145 IF C$>MA$ OR C$<MI$ THEN 140
146 PRINT C$;"_";CHR$(29);
147 LET D$=D$+C$
148 LET LO=LO+1
149 IF LO<>LO THEN GOTO 140
150 PRINT " ":GOSUB 123:RETURN
151 REM
152 REM ***********************
153 REM * SALIDA DEL PROGRAMA *
154 REM ***********************
155 REM
156 GOSUB 117:LOCATE 22,1:PRINT "PULSA [ESCAPE] PARA SALIR":PRINT "OTRA TECLA CO
NTINUAR":A$=INPUT$(1):IF ASC(A$)=27 THEN CLS:SYSTEM ELSE GOSUB 123:GOTO 74
157 REM
158 REM ***************
159 REM * DIRECTORIO *
160 REM ***************
161 REM
162 CLS:LOCATE 1,1:PRINT N$;" DE LA UNIDAD ";D$;":"
163 FILES D$+":*.*":PRINT:PRINT
164 RETURN
165 REM
166 REM **************************
167 REM * PEDIR UNIDAD DE DISCOS *
168 REM **************************
169 REM
170 PSET(X,Y),P:M$=N$+" DE LA UNIDAD = ":MA$="z":MI$="A":LO=2:GOSUB 129:IF LEN(D
$)=2 THEN 170 ELSE IF D$="" THEN D$="A"
171 RETURN
172 REM
173 REM *****************
174 REM * INSTRUCCIONES *
175 REM *****************
176 REM
177 GOSUB 105:CLS:LOCATE 1,1:PRINT "  COMANDOS DEL PROGRAMA SUPERGRAFICOS."
178 PRINT:PRINT "<Q> ...... Finaliza sesion"
179 PRINT "<P> ...... Dibuja un punto (PSET)"
180 PRINT "<O> ...... Borra un punto (PRESET)"
181 PRINT "<L> ...... Traza una linea (LINE)"
182 PRINT "<X> ...... Visualiza coordenadas"
183 PRINT "<M> ...... Graba pantalla (SAVE)"
184 PRINT "<N> ...... Carga pantalla (LOAD)"
185 PRINT "<K> ...... Graba pantalla en memoria"
186 PRINT "<J> ...... Carga pantalla de memoria"
187 PRINT "<ESC> .... Anula la ultima accion (DEL)"
188 PRINT "<Z> ...... Borra la pantalla (CLS)"
```

```
189 PRINT "<B> ...... Borra un fichero (KILL)"
190 PRINT "<D> ...... Muestra el directorio (FILES)"
191 PRINT "<R> ...... Renombra un fichero (RENAME)"
192 PRINT "<S> ...... Varia la velocidad del cursor"
193 PRINT "<C> ...... Cambia color de tinta (COLOR)"
194 PRINT "<F> ...... Colorea una zona (FILL)"
195 PRINT "<W> ...... Copia parte del dibujo (WIND)"
196 PRINT "<?> ...... Muestra esta pantalla"
197 PRINT "<2,4,6,8>  Mueven el cursor"
198 PRINT:PRINT "PULSA UNA TECLA";:A$=INPUT$(1):GOTO 111
199 REM
200 REM ***************************
201 REM * CREACION DE UNA VENTANA *
202 REM ***************************
203 REM
204 A$=INKEY$:IF A$="" THEN 204
205 IF A$=CHR$(13) THEN GOTO 74
206 IF MID$(A$,2,1)="K" THEN PUT (X,Y),E%,PSET:X=X-SS-SS*(X-SS<0):GOTO 211
207 IF MID$(A$,2,1)="M" THEN PUT (X,Y),E%,PSET:X=X+SS+SS*(X+SS>319*SW-AX):GOTO 2
11
208 IF MID$(A$,2,1)="H" THEN PUT (X,Y),E%,PSET:Y=Y-SS-SS*(Y-SS<0):GOTO 211
209 IF MID$(A$,2,1)="P" THEN PUT (X,Y),E%,PSET:Y=Y+SS+SS*(Y+SS>199-AY):GOTO 211
210 GOTO 204
211 GET (X,Y)-(X+AX,Y+AY),E%:PUT (X,Y),D%,PSET: GOTO 204
```

Al principio del programa, después de introducirlo y hacer RUN, el ordenador nos preguntará qué tipo de pantalla queremos utilizar. Caben dos posibles opciones:

— 320 o pantalla en color (4 colores).
— 640 o pantalla monocromo (tinta y fondo).

Elige la que quieras utilizar en cada momento. La diferencia entre las dos, aparte de la capacidad de color, es que con la primera tenemos 320 puntos de ancho y con la segunda 640.



Las funciones que es capaz de realizar este programa se manejan con una sola mano, ya que todas las opciones posibles se pueden realizar con sólo pulsar una tecla. Estas son:

P. Dibuja un punto en la posición actual del cursor gráfico.

O. Borra el punto sobre el que se encuentra el cursor gráfico (si lo hubiese).

L. Traza una línea desde el último punto que se dibujó hasta la posición actual del cursor gráfico. Si el punto fuese un punto en el color del papel (borrar) la línea será también del color del papel (borrará).

X. Visualiza en pantalla las coordenadas actuales del cursor gráfico.

M. Graba la pantalla en el disco. Después de pulsar esta tecla, se le pregunta al usuario con qué nombre se va a grabar. Si el nombre no tiene extensión (.BIN por ejemplo) el ordenador asignará àutomáticamente la extensión .BAS.

N. Carga una pantalla que se encuentre en disco. Después de pulsar esta tecla se le pregunta al usuario el nombre de la pantalla a cargar.

K. Este comando graba la pantalla en la memoria. Es un comando muy útil, pues nos permite hacer copias de seguridad de la pantalla sin tener que hacerlas en

el disco. Esto sirve para que, en el caso de borrar la pantalla por error, no perdamos lo que hemos hecho.

J. Recupera la pantalla que se almacenó en memoria con el comando K.

ESCAPE. Anula la última acción realizada. Si hemos trazado una línea equivocadamente, pulsando la tecla ESCAPE la pantalla volverá a aparecer como estaba antes de trazar dicha línea.

Z. Borra la pantalla. Antes de borrarla pide la confirmación del usuario.

B. Borra un fichero del disco. También saca el directorio para que el usuario vea los ficheros que contiene el disco.

D. Muestra el directorio. El programa pide la unidad de la cual se quiere conocer el directorio.

R. Renombra un fichero. Antes de hacerlo saca el directorio de la unidad que se le especifique.

S. Varía la velocidad con la que se mueve el cursor gráfico. Dicha velocidad se puede regular entre 1 y 9.

F. Colorea una zona cerrada, como un círculo, rellenándola de color.

C. Cambia el color de la tinta. El color se puede variar entre 0 y 3 en el modo de 320, y entre 0 y 1 en el modo de 640.

W. Nos permite copiar una parte de la pantalla a otro lugar de la misma. Para ello hace falta poner el cursor gráfico en la esquina superior izquierda del trozo a copiar y pulsar la letra W. Después nos movemos a la esquina inferior derecha y volvemos a pulsar la letra W. Una vez hecho esto, el cursor desaparecerá y nosotros podremos mover la zona elegida por la pantalla con las teclas del cursor. Una vez que esté donde nosotros queremos, pulsamos ENTER y el cursor volverá a aparecer.

Las teclas que mueven el cursor gráfico son las que se encuentran en el teclado numérico que están en la parte derecha del teclado. Estas teclas son:

2 — Movimiento hacia abajo.
8 — Movimiento hacia arriba.
4 — Movimiento hacia la izquierda.
6 — Movimiento hacia la derecha.

Si en cualquier momento de la ejecución del programa queremos ver todos los comandos de que dispone el SUPERGRAFICS, sólo tenemos que pulsar el interrogante (?) para que aparezcan en pantalla.



Notas sobre el programa 4

Para cargar las pantallas que realices, desde uno de tus programas, sólo tienes que poner las siguientes instrucciones:

```
DEF SEG=&HB800:BLOAD "nombre"
```





Comandos del programa Supergráficos.



Fichero de discos

Este último programa, también para el SPECTRUM, nos permitirá tener almacenada, de una forma cómoda y que no ocupa espacio, toda la información referente a nuestros libros. Con el programa podremos saber en cualquier momento quién escribió un cierto libro, qué editorial tiene en su catálogo un libro que nos interesa, en qué fecha lo compramos, etcétera.

```
FICHERO DE LIBROS
-----------------

 10 LET NF=1
 20 DIM T$(100,50)
 22 DIM A$(100,50)
 24 DIM C$(100,50)
 25 DIM O$(100,50)
 30 CLS
 35 PRINT  INVERSE 1;"=========
============================  FI
CHERO D E LIBROS  ==============
========================="
 40 PRINT AT 6,7; INVERSE 1;" 1
"; INVERSE 0;" FICHA NUEVA.";AT
```

```
 8,7; INVERSE 1;" 2 "; INVERSE 0
;" CARGAR FICHERO.";AT 10,7; INV
ERSE 1;" 3 "; INVERSE 0;" SALVAR
 FICHERO.";AT 12,7; INVERSE 1;"
4 "; INVERSE 0;" LISTAR FICHEROS
.";AT 14,7; INVERSE 1;" 5 "; INV
ERSE 0;" BUSCAR UNA FICHA.";AT 1
6,7; INVERSE 1;" 6 "; INVERSE 0;
" BORRAR EL FICHERO."
  42 PRINT AT 21,1; INVERSE 1;"
OPCION = "; FLASH 1;" ";CHR$ 8;
  50 LET K$=INKEY$
  60 IF K$<"1" OR K$>"6" THEN  G
O TO 50
  65 PRINT  INVERSE 1;K$;" "; OV
ER 1;AT 4+2*VAL K$,7;"
              ": FOR I=1 TO 200:
NEXT I
  70 GO TO 1000*VAL K$
1000 CLS : PRINT  INVERSE 1;"
  INTRODUCIR FICHA NUEVA      "
1001 PRINT  INVERSE 1;"=========
======================="
1002 PRINT  INVERSE 1;"      FICH
AS EN MEMORIA =          ";AT 2,25
;NF-1
1005 INPUT "TITULO: "; LINE T$(N
F)
1006 PRINT ''' INVERSE 1;" TITUL
LO........: "; INVERSE 0;" ";T$(
NF)
1010 INPUT "AUTOR: "; LINE A$(NF
)
1015 PRINT ' INVERSE 1;" AUTOR .
........: "; INVERSE 0;" ";A$(NF
)
1020 INPUT "CLASIFICACION: "; LI
NE C$(NF)
1025 PRINT ' INVERSE 1;" CLASIFI
CACION.: "; INVERSE 0;" ";C$(NF)
1030 INPUT "EDITORIAL: "; LINE O
$(NF)
1040 PRINT ' INVERSE 1;" EDITORI
AL.....: "; INVERSE 0;" ";O$(NF)
1050 LET NF=NF+1
1400 PRINT #0; INVERSE 1;" INTRO
DUCIR OTRA FICHA      S/N ";
1410 LET W$=INKEY$
1420 IF W$="S" OR W$="s" THEN  G
O TO 1000
1430 IF W$<>"N" AND W$<>"n" THEN
  GO TO 1410
1440 GO TO 30
2000 GO SUB 2400
2020 INPUT "CATALOGO ?(S/N)"; LI
NE B$
2030 IF B$="S" THEN  CAT 1
2040 GO SUB 2300
2060 LOAD *"M";1;N$+"A" DATA A$(
)
2061 LOAD *"M";1;N$+"T" DATA T$(
)
2062 LOAD *"M";1;N$+"C" DATA C$(
)
2063 REM  LOAD *"M";1;N$+"O" DAT
A O$()
2065 GO SUB 4400
2070 GO TO 30
2100 GO SUB 2300
2110 LOAD N$+"A" DATA A$()
2111 LOAD N$+"T" DATA T$()
2112 LOAD N$+"C" DATA C$()
2113 LOAD N$+"O" DATA O$()
2115 GO SUB 4400
2120 GO TO 30
2300 INPUT "NOMBRE DEL FICHERO ?
(MAX.9 C.)"; LINE N$
2310 IF LEN N$>9 THEN  GO TO 204
0
2320 RETURN
2400 INPUT "CASSETTE (C) O MICRO
 (M)"; LINE B$
2410 IF B$="C" THEN  GO TO VAL K
$*1E3+100
2420 IF B$<>"M" THEN  GO TO 2400
2430 RETURN
3000 GO SUB 2400
3010 GO SUB 2300
3020 SAVE *"M";1;N$+"A" DATA A$(
)
3021 VERIFY *"M";1;N$+"A" DATA A
$()
3022 SAVE *"M";1;N$+"T" DATA T$(
)
3023 VERIFY *"M";1;N$+"T" DATA T
$()
3024 SAVE *"M";1;N$+"C" DATA C$(
)
3025 VERIFY *"M";1;N$+"C" DATA C
$()
3026 REM  SAVE *"M";1;N$+"O" DAT
A O$()
3027 REM  VERIFY *"M";1;N$+"O" D
ATA O$()
3030 GO TO 30
3100 GO SUB 2300
3110 SAVE N$+"A" DATA A$()
3111 SAVE N$+"T" DATA T$()
3112 SAVE N$+"C" DATA C$()
3120 GO TO 30
4000 CLS : PRINT  INVERSE 1;"===
=============================
        LISTAR FICHERO          ===
=============================="
4001 PRINT '''"IMPRIMIMOS DESDE
LA FICHA ";: INPUT "NO. DE FICHA
 ";N1
4002 PRINT N1
4003 PRINT ''"HASTA LA FICHA ";:
 INPUT "NO DE FICHA ";N2
4004 PRINT N2
4005 IF (N1<1 OR N1>NF-1)  OR (N
2<N1  OR N2>NF-1) THEN  GO TO 40
00
4006 PRINT #0; INVERSE 1;" PULSA
 UNA TECLA ";
4009 FOR N=N1 TO N2
4010 LET B$=A$(N)
4020 GO SUB 4300
4030 LET D$=B$
4040 LET B$=T$(N)
4050 GO SUB 4300
4060 LET E$=B$
4070 LET B$=C$(N)
4080 GO SUB 4300
4090 LET F$=B$
4091 LET B$=O$(N)
4092 GO SUB 4300
4093 LET G$=B$
4096 CLS
4100 PRINT AT 0,0;" -FICHA ";N;"
-"
4110 PRINT AT 9,0; INVERSE 1;"TI
TULO : "; INVERSE 0;" ";E$
4120 PRINT AT 12,0; INVERSE 1;"A
UTOR : "; INVERSE 0;" ";D$
4130 PRINT AT 15,0; INVERSE 1;"C
LASIFICACION : "; INVERSE 0;" ";
F$
4132 PRINT AT 18,0; INVERSE 1;"E
DITORIAL "; INVERSE 0;" ";G$
4140 PRINT #0; FLASH 1;"PULSA UN
A TECLA. Q PARA IMPRIMIR"; FLASH
 0
4145 PAUSE 0: IF INKEY$="Q" THEN
  COPY
4150 NEXT N
4160 PAUSE 0
4170 GO TO 30
4300 FOR A=1 TO LEN B$+1
4310 IF B$(A)<>" " OR B$(A+1)<>"
 " OR B$(A+2)<>" " OR B$(A+3)<>"
 " THEN  NEXT A
4320 LET B$=B$(1 TO A-1)
4330 RETURN
4400 FOR A=1 TO 100
4410 IF A$(A,1)=" " THEN  LET NF
=A: RETURN
```

```
4420 NEXT A
4430 RETURN
5000 INPUT "TITULO QUE BUSCAS ?"
; LINE G$
5010 FOR B=1 TO NF-1
5015 LET B$=T$(B)
5016 GO SUB 4300
5020 LET N1=B: LET N2=B: GO TO 4
009
5030 NEXT B
5040 CLS
5050 PRINT "TITULO NO PRESENTE E
N EL FICHERO"
5060 PAUSE 0
5070 GO TO 30
6000 INPUT "NUMERO DE FICHA ?";F
6010 IF F>NF THEN  CLS : PRINT A
T 0,0;"NO HAY TANTAS FICHAS.": P
AUSE 0: GO TO 30
6020 FOR A=F TO NF+1
6030 LET A$(A)=A$(A+1)
6040 LET T$(A)=T$(A+1)
6050 LET C$(A)=C$(A+1)
6060 NEXT A
6070 LET NF=NF
6080 GO TO 30
```




Ejemplo de ficha.

El número de fichas que puede aceptar el programa es de 150. Si tenemos más libros no hay ningún problema. Los podemos agrupar por temas y crearnos un fichero distinto por cada tema.

Este programa aparecerá en las versiones para los distintos ordenadores en tomos sucesivos.



Ecualizador

Este programa es una pequeña curiosidad para los usuarios del SPECTRUM. Con él podrán ver en la pantalla del ordenador cómo es la onda sonora de la voz humana, de un ruido, de un instrumento, de una canción, etc.

FICHERO DE LIBROS

1 FICHA NUEVA.
2 CARGAR FICHERO.
3 SALVAR FICHERO.
4 LISTAR FICHEROS.
5 BUSCAR UNA FICHA.
6 BORRAR EL FICHERO.

OPCION =

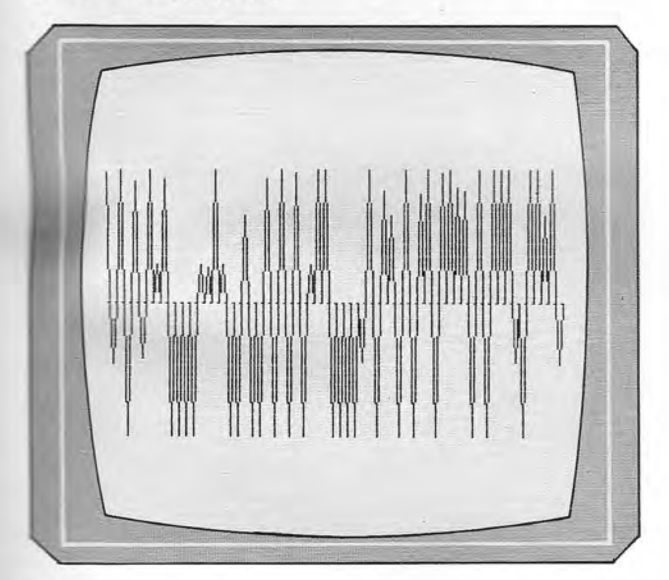

Menú del progama «Ficheros de libros».

```
ECUALIZADOR
-------------

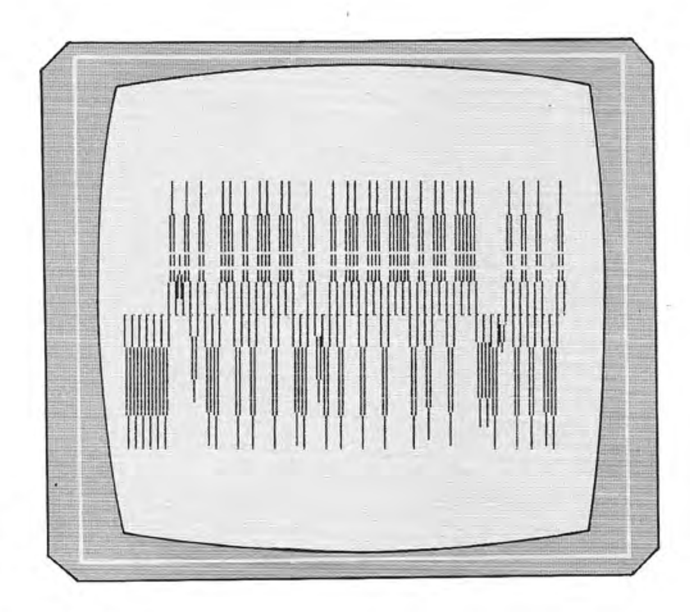
   1 REM ++++++++++++++++++
   2 REM + ECUALIZADOR DE +
   3 REM +    SONIDOS     +
   4 REM ++++++++++++++++++
   5 REM + POR:           +
   6 REM +  CARLOS CORAL  +
   7 REM ++++++++++++++++++
   8 REM
   9 GO SUB 100
  10 LET K$=INKEY$
  11 PRINT AT 10,9; INVERSE 1;"P
ULSA UNA TECLA"
  12 IF K$="" THEN  GO TO 10
  13 CLS : LET DIR=23300
  14 LET X=0
  15 PLOT X,87
  16 PLOT X+1,87
  17 LET Y=PEEK DIR
  18 IF Y>175 THEN  LET Y=175
  19 PLOT X+1,87
  20 PLOT X,87
  21 LET A=Y-87
  22 LET B=87-Y
  23 DRAW 2,A
  24 DRAW 2,B
  25 LET L=USR 64000
  26 LET X=X+4
```

```
 27 IF X>=252 THEN  CLS : LET X
=0
 28 GO TO 17
 100 FOR F=64000 TO 64040
 101    READ A
 102    POKE F,A
 103 NEXT F
 104 RETURN
 105 REM
 110 DATA 175,50,4,91,6,8,62,117
 111 DATA 219,254,203,119,204,21
,250,196
 112 DATA 31,250,16,242,201,58,4
,91,203,135
 113 DATA 23,50,4,91,201,58,4,91
,203,199
 114 DATA 23,50,4,91,201
```

Lo único que tienes que hacer para que este programa funcione es, aparte de teclearlo y hacer RUN, meter una cinta grabada en el cassette que utilizas para leer y grabar programas y pulsar la tecla PLAY. En cuando comience a sonar la cinta veremos en la pantalla la onda de dicho sonido.



Onda que realiza el programa Ecualizador cuando escucha en Rock and Roll.





Esta es la onda de la voz humana.



Notas al programa 6

Para que el programa entienda bien la música, o lo que esté grabando en la cinta, no es necesario poner muy alto el volumen. Es conveniente que éste se encuentre en la mitad de su recorrido.

TECNICAS DE ANALISIS

TABLAS DE DECISION



LAS tablas de decisión constituyen un procedimiento simple y claro de presentar las condiciones de proceso en el tratamiento de datos. Es normal que los diferentes procesos elementales puedan ser definidos individualmente, así como las condiciones bajo las que dichos procesos deben aplicarse. Una tabla de decisión es una matriz en la cual se intercalan estas condiciones que han de cumplirse para que se apliquen unos tratamientos dados y los tratamientos que han de aplicarse en cada caso.

En una tabla de decisión aparecen cuatro partes claramente diferenciadas:

1) CONDICIONES	3) SITUACIONES
2) TRATAMIENTOS	4) SECUENCIAS DE TRATAMIENTOS



Estructura de una tabla de decisión.

1) Matriz de condiciones.

En ella aparecen todas y cada una de las condiciones que intervienen en el proceso estudiado, al cual se refiere la tabla de decisión.

Es importante que estas condiciones estén perfectamente definidas y sean independientes una de otras (es decir, se puedan identificar sin interrelación con las restantes): la tabla de decisión, precisamente, establecerá la relación de unas con otras y con los tratamientos a aplicar en cada caso.

2) Matriz de tratamientos.

Aparecen en esta sección de la tabla de decisión todos los diversos tratamientos que habrá que aplicar en las diferentes situaciones del proceso que se estudia.

Cada uno de estos tratamientos representará un pequeño módulo de proceso dentro del pograma general en que se plasmará la tabla de decisión.

3) Situaciones.

Conjunto de posibles combinaciones de las condiciones descritas en la matriz de la izquierda de la tabla de decisión.

En esta parte de la tabla se van anotando, por columnas, las condiciones que deben darse en cada una de las posibles situaciones.

4) Secuencias de tratamientos.

Combinaciones de los distintos tratamientos a aplicar en función de las situaciones diferentes que aparecen en la parte tercera de la tabla.

Para cada situación se reseña, en la

misma columna, el conjunto de los tratamientos a aplicar a los datos en esa situación.

	Situaciones						
Condición 1	S	N	S	S	S		S
Condición 2	N	N	S	S	N		N
Condic. 3 caso A	S	S	S	S	N		S
Condic. 3 caso B	N	N	N	S	N	etc.	S
Condic. 3 caso C	N	N	S	S	S		S
Condición 4	S	S	S	N	N		S
.							
Condición m	S	S	N	N	N		N
Tratamiento 1	X	X		X			X
Tratamiento 2		X	X	X	X		
Tratamiento 3 (ir a tabla B)	X			X			X
.							
.						etc.	
.							
Tratamiento m		X			X		X



Un ejemplo de tabla de decisión.

De acuerdo con lo anterior, se puede ver en la figura 2 un ejemplo de tabla de decisión: la primera situación se producirá cuando SI se cumple la condición 1, pero NO la 2; de la condición 3, SI se cumple el caso A pero NO los casos B y C, etc. La prolongación hacia abajo de esa columna (correspondiente a la situación 1.º) nos indica que en este caso se debe aplicar el tratamiento 1 y el tratamiento 3 (que conduce, a su vez, a otros procesos o tratamientos descritos en otra tabla de decisión: la tabla B). Del mismo modo, la situación segunda (caso en que SI se cumplen las condiciones 3-caso A, 4,... m; y no se cumplen las restantes condiciones) exige que se apliquen los tratamientos 1, 2,... m.

Existen tablas de tipo «ampliado»: cuando las condiciones presentan más de dos alternativas o conjunto de valores. En contraposición a ellas, cuando en una tabla de decisión todas las condiciones son simples, la tabla se suele llamar «limitada». La tabla presentada en la figura 2 es del tipo «mixto», pues aparecen en ella condiciones simples y una condición (la tercera) con varias alternativas. A veces, las tablas de decisión ampliadas o mixtas presentan en una sola condición las diferentes opciones que se pueden dar; en ese caso se pueden convertir (como se ha hecho en la tabla de la figura 2) a tablas con condiciones simples desglosando en varias la condición múltiple.

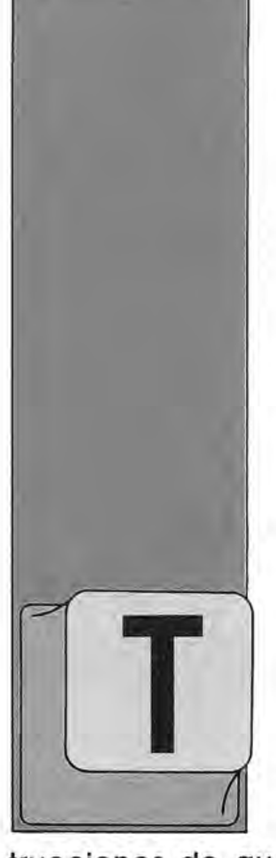
Por otro lado, se suelen distinguir también, en las tablas de decisión, aquéllas que son completas en sí mismas (tablas «cerradas») de las que (como el ejemplo presentado) remiten, en algunos de los tratamientos propuestos, a la consulta de otra u otras tablas adicionales: estas tablas se suelen llamar «abiertas» o «recurrentes».

Otro aspecto a tener en cuenta es el tipo de reglas de decisión que se incluye en la tabla. Pueden ser de tipo AND, OR, ELSE o múltiples. La regla AND significa que han de darse todas las condiciones indicadas en la columna correspondiente de la matriz de situaciones, para que se ejecuten los tratamientos marcados en la parte inferior de la columna: como se está afirmando que ha de cumplirse la condición «i» Y la condición «j» Y la condición «K» Y... se designa este tipo de regla con el nombre inglés del operador lógico correspondiente (AND).

En el ejemplo propuesto en la figura 2 hemos considerado que las reglas de decisión (columnas de la matriz de situaciones) son de tipo AND.

En las reglas de tipo OR se supone que se produce la situación descrita en la correspondiente columna (y que se deben aplicar, en consecuencia, los tratamientos indicados) si se cumple la condición «i» O la condición «j» O la condición «k» O etc.: es decir, se «disparará» la correspondiente regla si se cumplen una o más de las condiciones marcadas.

La regla de tipo ELSE se llama así en relación con la instrucción (común a varios lenguajes de programación) IF... ELSE... (SI... EN-OTRO-CASO...; SI sucede una condición, hacer esto; EN-OTRO-CASO hacer tal otra cosa); aparece este tipo de regla de decisión cuando sólo algunas de las posibles situaciones nos conducen a tratamientos especializados, mientras que todo el resto de las situaciones producen un tratamiento concreto (único para todas las restantes); en este caso, se detallan las situaciones que conducen a tratamientos específicos y el resto de las posibilidades se agrupan en una única regla de decisión (que se llama regla ELSE).



TECNICAS DE PROGRAMACION

ESTRUCTURAS DE DATOS

TRADICIONALMENTE se ha considerado que un programa de ordenador bien construido debe constar de dos partes claramente diferenciadas: por un lado, las instrucciones de que consta el programa; por otro, los datos sobre los que éstas actúan. A su vez, las instrucciones se dividen en cuatro grupos principales:

• Asignación a una variable del valor de una expresión.

• Estructuras de control, que permiten dirigir la ejecución del programa por los pasos necesarios para realizar la tarea propuesta.

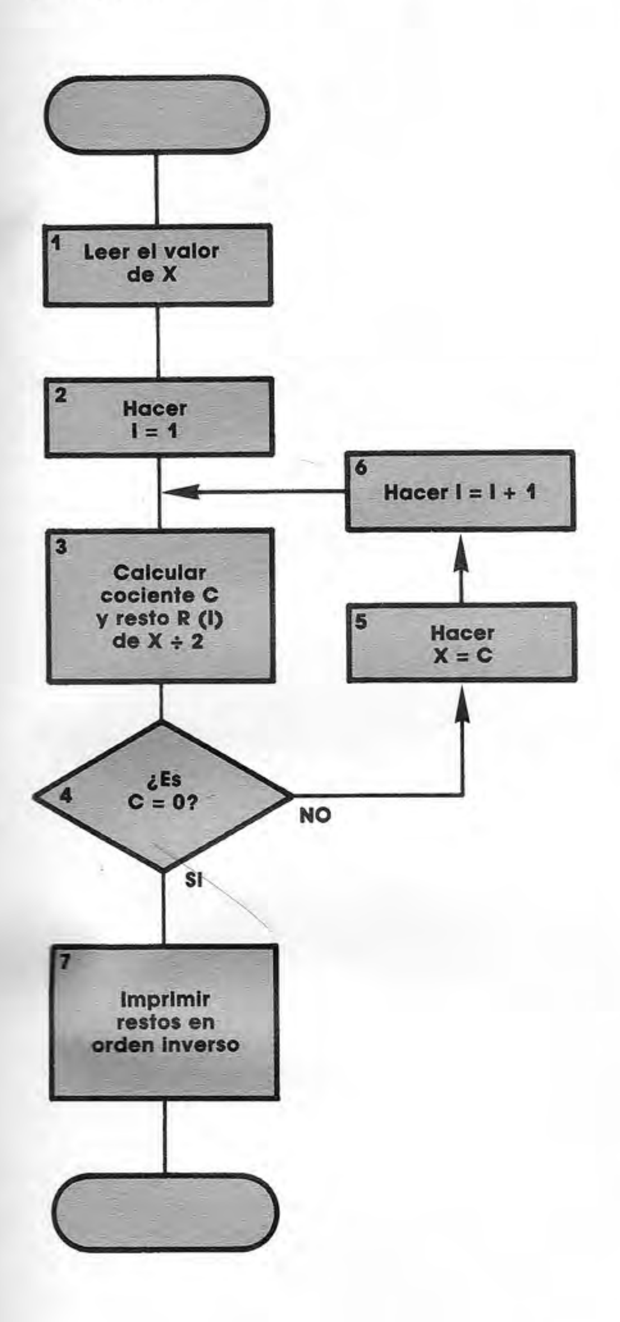
• Instrucciones de modularización, que descomponen un programa complejo en partes más pequeñas, cerradas en sí mismas y más fácilmente comprensibles, lo que facilita la construcción, mantenimiento y legibilidad de los programas obtenidos.

• Instrucciones de acceso a los dispositivos periféricos (entrada/salida), que permiten que nuestro programa pueda utilizar adecuadamente la pantalla, los ficheros en disco, la impresora, los gráficos y muchos otros dispositivos.

Pero un pograma sin datos no serviría para nada. Porque, en definitiva, ¿qué es un programa, sino un medio que, a partir de ciertos datos iniciales, nos permite calcular ciertos resultados que deseábamos conocer? Esta definición se aplica incluso a programas que aparentemente no calculan ningún dato concreto. Como ejemplo, pensemos en el control por ordenador de una máquina que ha de realizar un trabajo determinado: la fabricación de un tornillo o el ensamblaje de dos piezas en una cadena de montaje. Ante la mirada del observador casual, el programa que controla la máquina no parece estar calculando resultados de ningún tipo, pues se limita a regular la acción de la máquina y a dirigir sus mecanismos. Sin embargo, en realidad existe un considerable flujo de información, de datos, que van introduciéndose en la máquina por diversos medios y convirtiéndose en otros a través de cálculos más o menos complejos. Antes de poder mover el brazo de un robot que debe colocar cierta pieza en el lugar deseado, el programa debe poseer información sobre la posición actual de la pieza, el lugar exacto a donde debe llevarla y la localización de las distintas partes del brazo mecánico. Estos datos, que puede conseguir por medio de sensores visuales o táctiles, serán elaborados para obtener otros, que definen con todo detalle cada uno de los movimientos elementales que habrán de realizarse. Los resultados del cálculo son, también en este caso, datos, aunque no podamos verlos explícitos sobre el papel de una impresora.

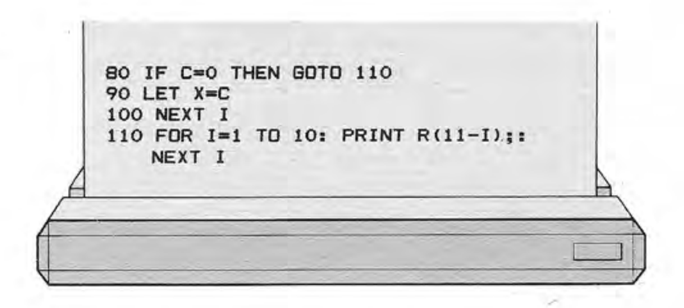
Veamos un ejemplo. En el capítulo primero vimos un algoritmo que nos permitía convertir a la base 2 un número entero cualquiera expresado en la base diez.

Repitamos aquí el algoritmo, en forma de organigrama:



Ahora vamos a ver cómo se construiría un programa que realice este algoritmo en el lenguaje BASIC.

```
10 DIM R(10)
20 FOR I=1 TO 10: LET R(I)=0: NEXT I
30 PRINT "dame un número entre
   0 y 1000"
40 INPUT X
50 FOR I=1 TO 10
60 LET C=INT(X/2)
70 LET R(I)=X-2*C
```



```
80 IF C=0 THEN GOTO 110
90 LET X=C
100 NEXT I
110 FOR I=1 TO 10: PRINT R(11-I);:
    NEXT I
```

Comparando el programa con el organigrama, podemos descubrir la siguiente correspondencia entre los bloques y las instrucciones:

Bloque n.°	Líneas n.°
1	30-40
2	50
3	60-70
4	80
5	90
6	100
7	110

En cuanto a las líneas 10 y 20 del programa, definen la variable R, donde se guardará la sucesión de restos (es decir, las cifras en base 2) y asignan un valor inicial de cero a cada uno de los diez lugares que tiene previstos para las cifras del resultado.

Obsérvese también que, en la instrucción 110, los valores de los restos se imprimen en orden inverso, como es preciso hacer para que las cifras del número convertido a la base 2 aparezcan en el orden correcto.

Siempre es difícil, en los programas escritos en BASIC, distinguir claramente entre las instrucciones y los datos, pues este lenguaje es muy poco estructurado. En particular, la regla que permite utilizar una variable sin haberla definido previamente facilita que los datos queden esparcidos a lo largo y a lo ancho de los programas y mezclados íntimamente con las instrucciones ejecutables.

En el caso del programa 1, los datos son los siguientes:

• La variable X, donde se guarda inicialmente el valor en base 10 que se desea convertir a la base 2, aunque a lo largo del programa este valor va siendo sustituido por los cocientes sucesivos, de acuerdo con el algoritmo.

• La varible C, donde se guarda el valor del cociente entero de X al dividirlo entre 2.

• La variable R, que no es como las anteriores, que tienen un solo valor, sino que puede considerarse como un agregado de diez valores diferentes, donde van a guardarse los restos sucesivos que vayamos obteniendo. Esto se debe a que no debemos perder estos restos, puesto que entre todos constituyen el resultado del programa. Podemos representar los diez valores distintos de la variable R con los símbolos R(1), R(2), R(3), R(4), R(5), R(6), R(7), R(8), R(9) y R(10).

• La variable I, que actúa como contador de los tres bucles de que consta el programa. Sólo uno de estos bucles (el central) aparece en el organigrama. Los otros dos, el de inicialización de los diez valores de R (instrucción 20) y el de escritura de los mismos (instrucción 110), están implícitos.

Puede verse que sólo la variable R está claramente separada de la parte ejecutable del programa. Esto se debe a que, tratándose de una variable especial (un agregado de datos), debe definírsela explícitamente como tal. La instrucción 10 tiene por objeto, precisamente, comunicarle este hecho al intérprete de BASIC, es decir, pedirle que le dé un tratamiento diferente al de las restantes variables DIM R(10) significa, pues, que R es una variable de «dimensión» 10, es decir, un agregado de diez valores.

El lenguaje BASIC permite colocar las instrucciones DIM en cualquier parte del programa, siempre que sea antes de la primera utilización de las variables correspondientes. Sin embargo, no es una buena técnica de programacion aprovecharse de esto para distribuir alegremente las instrucciones DIM por diversos lugares de nuestros programas, pues esto tiende a hacerlos menos legibles, lo que dificulta su comprensión posterior por otras personas, o incluso por nosotros mismos. Lo recomendable es colocar todas las instrucciones DIM al principio del programa, de manera que sea muy fácil ver, de una sola ojeada, cuáles son las variables que actúan como agregados de datos. Tan sólo existe una excepción a esta regla de buen programar, que se verá cuando hablemos de la modularización de los programas.

Las instrucciones del tipo de la línea 10, que no realizan ejecución alguna, sino que sólo siven para definir la estructura de ciertos datos que van a ser utilizados posteriormente por un programa, se llaman «instrucciones declarativas».

Las variables que podríamos llamar «normales» (que tienen un solo valor), como I, X y C en el ejemplo anterior, no es preciso definirlas, por lo que aparecerán por primera vez en nuestros programas BASIC cuando tengamos que hacer uso de ellas, y no antes. En este caso concreto es fácil localizarlas, pues el programa es corto. Así vemos que I aparece por primera vez en la línea 20, X en la 40 y C en la 60. Pero es fácil comprender que, si el programa BASIC es largo y complejo, será mucho más difícil separar los datos de las instrucciones.

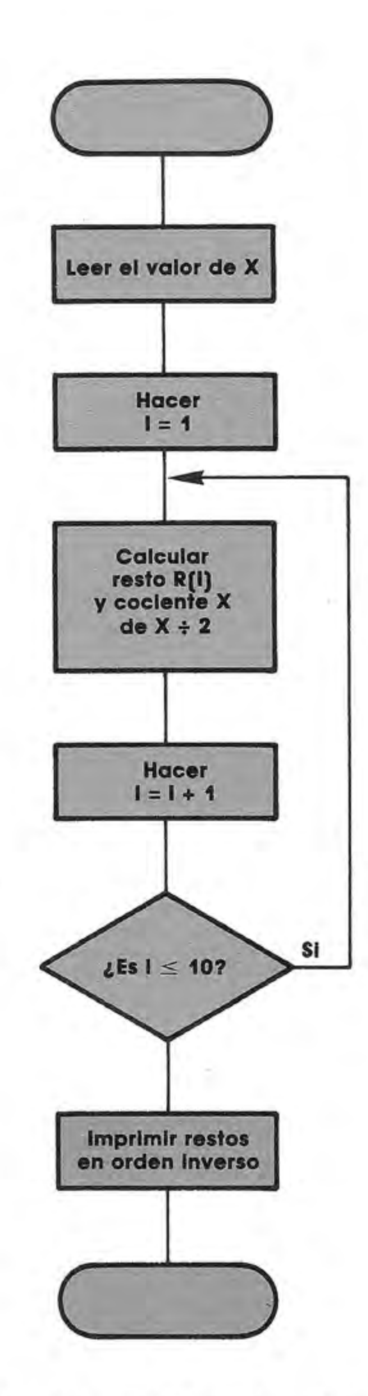
Si el programa anterior hubiera sido escrito en lenguaje PASCAL, no sería tan difícil distinguir las dos partes principales de todo programa: datos e instrucciones. En efecto: en PASCAL, todo programa debe comenzar por un conjunto de instrucciones declarativas, donde deben definirse todas y cada una de las variables que van a tomar parte en el programa, y que deben estar situadas al principio, antes de las instrucciones ejecutables. Esta es una de las razones por las que PASCAL se considera un lenguaje mucho más estructurado que BASIC, aunque hay otras, que iremos viendo posteriormente.

Veamos cómo se escribiría el mismo programa (a partir del mismo organigrama) en el lenguaje PASCAL.

Se puede ver cómo al principio de este programa aparecen, encabezadas por

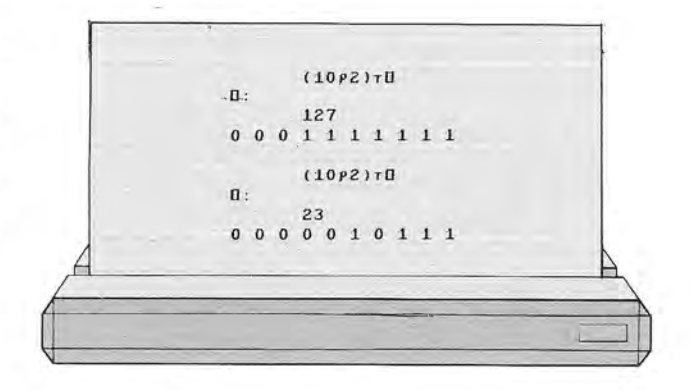
```
var
  x, i: integer;
  r: array[1..10] of integer;

begin
  writeln('Dame un número entre 0 y
  1000'); readln(x);
      for i:=1 to 10 do
        begin
          r[i]:= x mod 2;
          x:=x div 2;
        end;
      for i:=1 to 10 do write(r[11-i]);
end.
```



la palabra reservada «var», las declaraciones de todas las variables que utiliza: **x,** donde se guarda el número decimal entero que vamos a convertir a la base 2; **I,** que sirve como contador; y **r,** que es un agregado de diez valores, donde iremos colocando los restos sucesivos. En este caso no hemos inicializado los restos a cero, ni tampoco hemos abandonado el cálculo tan pronto se obtuvo el cociente cero, sino que continuamos calculando restos hasta que el contador del bucle (la variable I) alcanza el valor 10. Sin embargo, el resultado es totalmente equivalente al del programa anterior, pues todos los restos adicionales que obtendremos serán siempre iguales a cero. Debido a estas pequeñas modificaciones, el organigrama correspondiente a este programa no es exactamente igual al de la figura 1, sino que corresponde al de la figura 2.

Por último, puede ocurrir, como en el caso de un lenguaje de quinta generación como el APL, que la potencia del lenguaje sea tan grande que un organigrama relativamente detallado como el anterior resulte innecesario, pues es posible resolver el problema de la conversión de números de la base 10 a la base 2 en una sola instrucción ejecutable. En este caso, tampoco será necesario separar las instrucciones de los datos, pues el programa es tan reducido que la separación es obvia, y no precisa estructuración alguna. Veamos, por consiguiente, en el programa 3, cuál es esa instrucción APL que resuelve el problema de la conversión numérica, junto con un par de ejemplos de su utilización:



En este programa no existe ningún nombre de variable, pues no es necesario. El número decimal que deseamos convertir a la base 2 se introduce directamente desde el teclado (el símbolo cuadrado, en APL, significa que queremos obtener datos del teclado, y equivale hasta cierto punto a la instrucción INPUT de BASIC) y se convierte a la base deseada en una sola operación, representada por el símbolo semejante a una T pequeña. En cuanto a los datos incluidos entre paréntesis a la izquierda de este símbolo, el 2 indica la base de numeración a la que se desea convertir y el 10 es el número de cifras que deseamos obtener del número binario resultante.

Como es lógico, el organigrama correspondiente a este programa no es tampoco el de la figura 1, sino mucho más sencillo. Veámoslo en la figura 3.

Para terminar, hay que añadir que cualquiera de los programas anteriores puede transformarse fácilmente en otro que realice la conversión de la base 10 a otra base calquiera distinta de la base 2. Para ello, en el caso de la versión BASIC, bastará cambiar el 2 que aparece en las líneas 60 y 70 por la base de que se trate. Lo mismo ocurre con el programa PASCAL, donde también aparece la base 2 dos veces, en dos instrucciones consecutivas. En el caso del programa APL, también hay que sustituir el 2 incluido entre paréntesis por la base de numeración deseada.





PASCAL

DEFINICION DE DATOS EN UN PROGRAMA PASCAL

VEAMOS el siguiente programa.

```
program Calculador;
begin
  writeln ('El número ',5,
  'al cuadrado vale ',5*5);
  writeln ('El número ',5,
  'al cubo vale ',5*5*5)
end.
```

Como veremos más adelante con mayor detalle, la instrucción «writeln» sirve para sacar por pantalla las frases, números y resultados de operaciones matemáticas que aparecen entre paréntesis justo detrás de ella. Por tanto, en la pantalla aparecerá algo así:

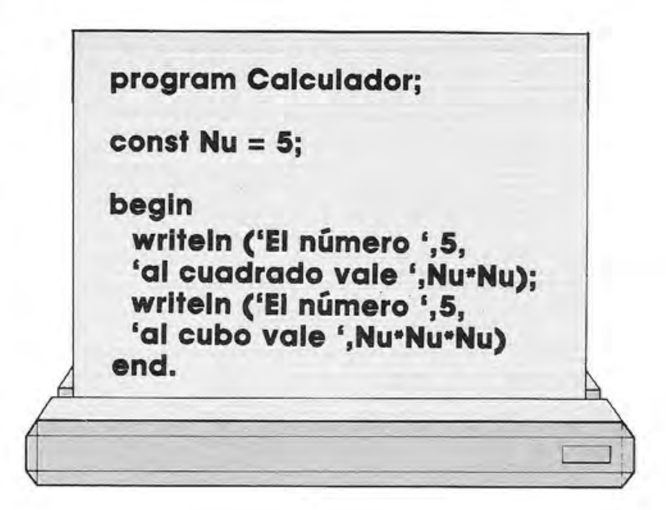
```
El número 5 al cuadrado vale 25.
El número 5 al cuadrado vale 125.
```

Si ahora quisiéramos cambiar el programa para que hiciese los cálculos con, por ejemplo, 6 en lugar de 5, tendríamos que sustituir todos los cincos que aparecen en el programa por seises. Si el programa fuese largo, esto podría llegar a ser muy tedioso.



Constantes numéricas

En PASCAL es posible asociar nombres (identificadores) a los números que van a ser constantes a lo largo del programa. Este nombre será luego el que se emplee en lugar del número. La asociación se hace en la zona de descripción de datos (es decir, tras la cabecera del programa); si lo empleamos en el programa anterior:



```
program Calculador;

const Nu = 5;

begin
  writeln ('El número ',5,
  'al cuadrado vale ',Nu*Nu);
  writeln ('El número ',5,
  'al cubo vale ',Nu*Nu*Nu)
end.
```

Este programa es totalmente equivalente al primero, pero hemos utilizado lo que se denomina «declaración de constantes».

Gracias a esto, cada vez que aparezca la palábra Nu en alguna parte del programa, será como si realmente hubiésemos puesto un cinco.

Si ahora quisiéramos que el programa funcionase con un seis, bastaría con retocar el sitio donde hemos dicho que Nu equivale a 5, es decir, pondríamos:

```
const Nu = 6;
```

La declaración de constantes siempre va precedida por la palabra reservada CONST y sirve para indicar al compilador que todo lo que venga a continuación, hasta que aparezca otra palabra reservada (que en este caso es BEGIN), son definiciones de constantes. Tras CONST se escribe el nombre a utilizar, un signo igual y el valor que queremos asociar al nombre. Al final de la zona de declaración de constantes siempre va un punto y coma (que, como iremos viendo, es lo que se utiliza más a menudo para separar cosas en PASCAL).

Podemos declarar todas las constantes que queramos sin más que poner sus definiciones una detrás de otra separadas entre sí por punto y coma:

```
const Juan = -15; Luis = 1234;
      Ene = -347;
```

Quedaría más claro así:

```
const
   Juan = -15;
   Luis = 1234;
   Ene = -347;
```

De esta forma, cada vez que en el programa aparezca Juan será como si hubiésemos puesto −15, y análogamente con los otros.



Textos

En el programa hay también textos o frases. En PASCAL las frases se escriben siempre entre apóstrofos como en «El número» o «al cubo vale». Dentro de una frase pueden ponerse letras, cifras, símbolos, espacios en blanco, etc., con total libertad; el compilador la toma tal como es, sin intentar analizar lo que contiene:

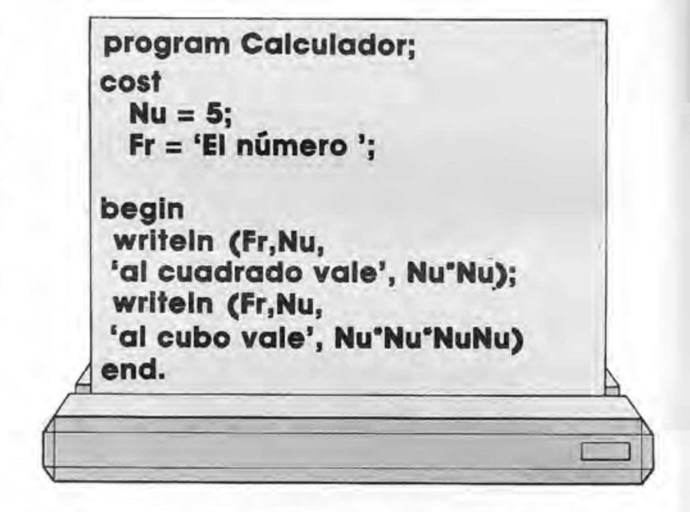
```
'Pone const y no pasa nada'
```

Para que aparezca un apóstrofe en medio de una frase sin que el compilador se crea que es el que marca el final, se pone por duplicado:

```
'Esta frase tiene un '' en medio'
```

Para que aparezca un apóstrofo en talla sólo aparecerá un apóstrofo.

Al igual que con los números constantes, podemos asociar un identificador a una frase. Es simplemente otro tipo de constante y por ello se declara de manera análoga:



Así, cada vez que aparece Fr es como si hubiésemos puesto realmente la frase.

Las frases son distintas de los números. Estos se guardan en la memoria del ordenador usando unos códigos especiales para ello. Sin embargo, si declaramos la constante C = '−54', ésta se guarda con un código para el signo menos, otro para la cifra 5 y otro para la cifra 4. C no se puede emplear para cálculos matemáticos; es sólo una frase que casualmente tiene aspecto de número.



Variables

Si los programas sólo pudieran trabajar con datos que son siempre los mismos,

servirían para bastante poco. En PASCAL, como es lógico, se pueden tener datos variables, pero hay que declararlos al principio del programa.

Los datos variables se guardan en porciones de la memoria del ordenador, siendo estas porciones de mayor o menor tamaño según el tipo de dato que tengan que alojar.

La declaración de variables sirve fundamentalmente para que:

1. El compilador reserve las porciones de memoria necesarias y del tamaño adecuado.
2. Asociar a cada porción un nombre para así poder utilizar el dato que guarde llamándole por ese nombre.

La zona de declaración de variables se encuentra justo tras la de las constantes (caso de existir ésta) y antes de la zona de instrucciones. Por tanto, la estructura de un programa queda de la siguiente manera:



El comienzo de la zona de definición de variables se indica con la palabra reservada VAR, tras la que se escriben todas las definiciones que se necesiten separadas entre sí por punto y coma. Cada definición consta del nombre de la variable seguido de dos puntos y de la descripción del tipo de variable a que corresponde. Veamos un ejemplo:

```
var
   Edad   : integer;
   Peso   : integer;
   Inicial : char;
```

Cuando dos o más variables son del mismo tipo, se puede indicar de manera resumida poniendo sus nombres uno a continuación de otro y separados por comas:

```
var
   Edad, Peso : integer;
   Inicial    : char;
```

INTEGER y CHAR son dos tipos de datos predefinidos que sirven, respectivamente, para guardar números enteros y caracteres (es decir, textos de un solo carácter).

Por tanto, estaríamos indicando al compilador que queremos tener dos variables de nombre «Edad» y «Peso» para guardar números enteros y otra de nombre «Inicial» para guardar caracteres.

En PASCAL existen muchos tipos de datos además de INTEGER y CHAR, e incluso es posible inventar nuevos tipos. Una regla de oro del PASCAL es la siguiente:

Nunca se pueden mezclar entre si tipos distintos de datos

El compilador, al traducir nuestro programa, vigila que no intentemos guardar, por ejemplo, un número entero en una variable de tipo CHAR, y avisa en su caso de la presencia de un error.

Otra posibilidad muy interesante es la de limitar los valores que pueda tomar una variable. Así, si intentáramos guardar un valor fuera de los límites establecidos, se produciría un error. Para hacer esto, en lugar del nombre del tipo se escriben el valor límite inferior y el superior separados entre sí por un par de puntos:

```
var
   Edad   : 0..110;
   Peso   : 0..200;
   Inicial : 'A'..'Z';
```

De esta manera, Edad sólo podría guardar números entre 0 y 110 e Inicial sólo letras de la A a la Z.

Por supuesto, podríamos utilizar constantes declaradas previamente para definir los límites:

```
const
   EdadMaxima = 110;
   PesoMaximo = 120;

var
   Edad   : 0..EdadMaxima;
   Peso   : 0..PesoMaximo;
   Inicial : 'A'..'Z';
```

NOTA: No con todos los compiladores se produce siempre el control de límites; a menudo es opcional. Por otra parte, a veces se consigue una disminución de la cantidad de memoria necesaria para una variable al limitar sus valores.



Instrucciones de programa

Una vez que tenemos declaradas las constantes y variables que vamos a utilizar, hay que indicar qué es lo que queremos hacer, y eso se consigue por medio de instrucciones que se escriben una detrás de otra, separadas entre sí por punto y coma, tras la zona de declaración de datos y entre las palabras reservadas BEGIN y END (con punto final, no lo olvidemos).

Hay dos tipos básicos de instrucciones:

— Las que hacen algo con los datos: leerlos de teclado, sacarlos por la pantalla o impresora, generar nuevos datos a partir de otros, etc.

— Las que sirven para regular cuáles, cuándo y cuántas veces ejecutar las instrucciones del grupo anterior.

Vamos a empezar por las instrucciones que sirven para mostrar datos.



Instrucciones de salida

Si quisiésemos mostrar por la pantalla el número −17, como internamente se guarda en la memoria usando un código especial, lo primero que tendríamos que hacer es deducir qué caracteres son los que hay que mostrar para que nosotros nos enteremos (un signo menos, un uno y un siete); a continuación habría que manejar la parte del ordenador encargada del control de la pantalla para que los presentase.

Afortunadamente, en PASCAL existen unos «procedimientos» o conjuntos de instrucciones ya preparados que hacen todo esto, de manera que no hay que preocuparse de esas cuestiones y basta con utilizar el nombre del procedimiento.

Uno que ya conocemos es WRITELN. Para utilzarlo, se escribe su nombre seguido de la lista de las diferentes cosas que queremos que aparezcan, separadas entre sí por comas; la lista debe ir entre paréntesis:

```
program Escritor;

   const Ene = -17;

begin
  writeln ('Esto es una frase.',
  Ene, '/Esto es otra. ',2*Ene)
end.
```

Este programa sacará por la pantalla:

```
Esto es una frase. -17/ Esto es otra. -34
```

En general, el tipo de cosas que se pueden mostrar son:

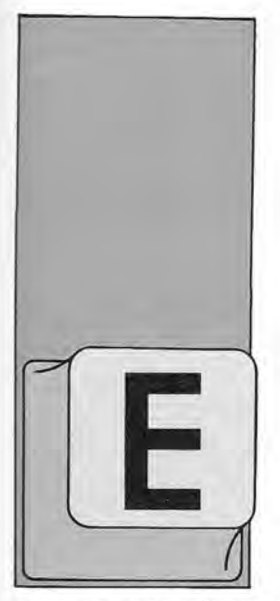
— Constantes numéricas o textos: writeln (14, 'Avión')

— Constantes declaradas: writeln (Ene)

— El valor de una variable: writeln (Peso)

— El resultado de cálculos matemáticos: writeln (2 * Ene)

HISTORIA DEL LENGUAJE C



EL lenguaje de programación C fue desarrollado por Dennis Ritchie, de los Laboratorios Bell, en la década de los setenta, cuando trabajaba, junto con Ken Thompson, en el diseño del sistema operativo UNIX.

El UNIX surgió por la necesidad de disponer de un sistema operativo potente que pudiese funcionar en pequeños ordenadores de propósito general. Esto dio lugar a que Ken Thompson desarrollara una primera versión de un pequeño sistema operativo, al que denominó UNIX, escrito en lenguaje ensamblador, siendo revisado posteriormente esta primera versión en un nuevo lenguaje de programación «lenguaje B», desarrollado a su vez por Ken Thompson.

El lenguaje B fue ampliado por Dennis Ritchie, el cual obtuvo una nueva versión a la que llamó «lenguaje C».

El momento actual del hardware, con el desarrollo de microordenadores de 16 bits que poseen la misma potencia que los miniordenadores de hace unos años, ha favorecido el uso del sistema operativo UNIX y el lenguaje C.

El lenguaje C se está transformando a pasos gigantescos en una de las bases de programación más importantes y populares.





C ha adquirido una amplia difusión.



Características fundamentales de C

El C es hoy día el lenguaje por excelencia de los miniordenadores que trabajan bajo el sistema operativo UNIX.

Pero no solamente se trabaja con C en miniordenadores y grandes sistemas. Actualmente se dispone en el mercado de compiladores del lenguaje C para ordenadores personales.





C puede utilizarse en ordenadores personales.

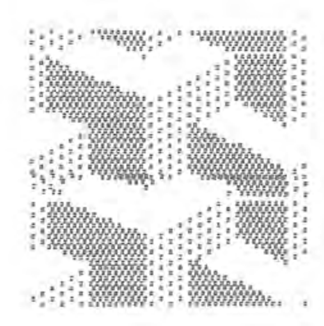
C es un lenguaje de programación de propósito general que produce «programas compactos» y «eficientes».





Los programas escritos en C ocupan menos espacio.

El diseño de programas en C puede llevarse a cabo bajo técnicas de programación estructurada (diseño "topdown")



C posee las estructuras básicas de la programación estructurada.

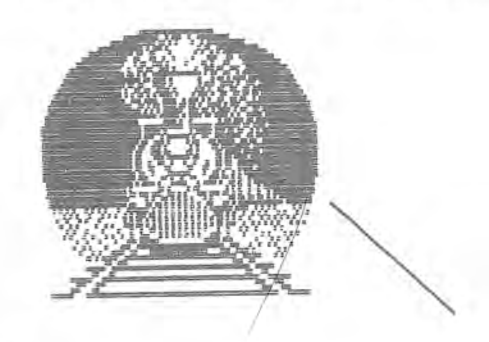
incorporando estructuras de control básicas como DO, UNTIL, WHILE, SWITCH e IF-THEN-ELSE. Los aficionados a la programación reconocerán este tipo de estructuras comunes, en gran medida, a otros lenguajes tales como el Pascal.





C es fácil de aprender.

C permite crear programas fáciles de modificar y de adaptar a nuevos ordenadores. Su «portabilidad» le hace el lenguaje idóneo para que programas en C escritos en un sistema puedan ejecutarse en otros con modificaciones mínimas.





C es portátil.

Hoy día existen compiladores C para unos 40 sistemas, abarcando desde microprocesadores de 8 bits hasta el superrápido Cray1.

Otra característica a resaltar del C es su «potencia» y «flexibilidad». El sistema operativo UNIX esta escrito, en su mayor parte, en lenguaje C. También existen cantidad de compiladores en el mercado desarrollados bajo C , entre los que podríamos mencionar Pascal, Lisp, Logo, Basic, etc.





C es potente.





C es flexible.





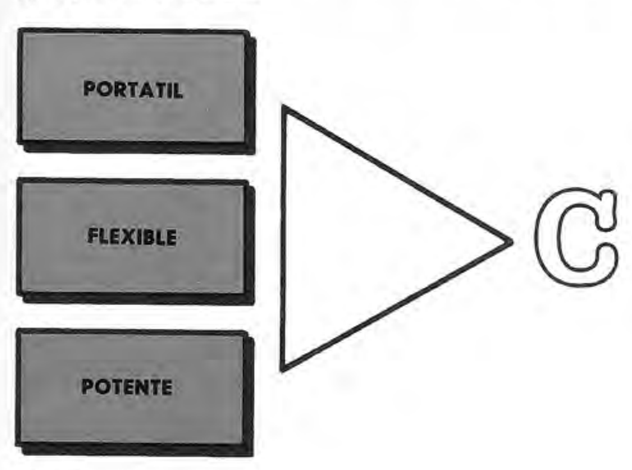
El sistema operativo UNIX está escrito en C.

Para los estudiosos de los diferentes lenguajes de programación diremos que C posee ciertas estructuras afines al Pascal, la sencillez del Basic y en ciertos aspectos llega a niveles tan bajos de la máquina que podemos decir que estamos trabajando con ensamblador (¡casi nada!).

En resumen, las características principales a destacar del lenguaje C son:

— Producir programas reducidos (código compacto).
— Producir programas eficientes.
— Ser un lenguaje portátil, potente y flexible.

— Servir de base para el desarrollo de compiladores de otros lenguajes de programación.
— Aprovechar las ventajas de la programación estructurada.
— Ser de gran utilidad para estudiantes y profesionales.
— La instalación es económica.
— Herramienta óptima para el diseño.
— Utilización en áreas tan extensas como los paquetes software, la gestión, los minis y microordenadores y los juegos, entre otros.



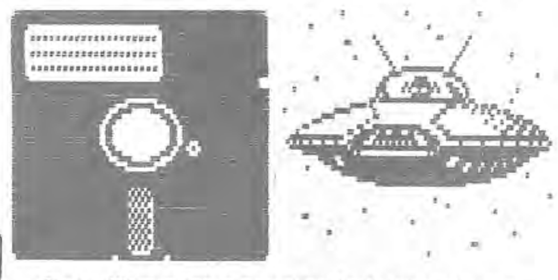

Algunas características de C.



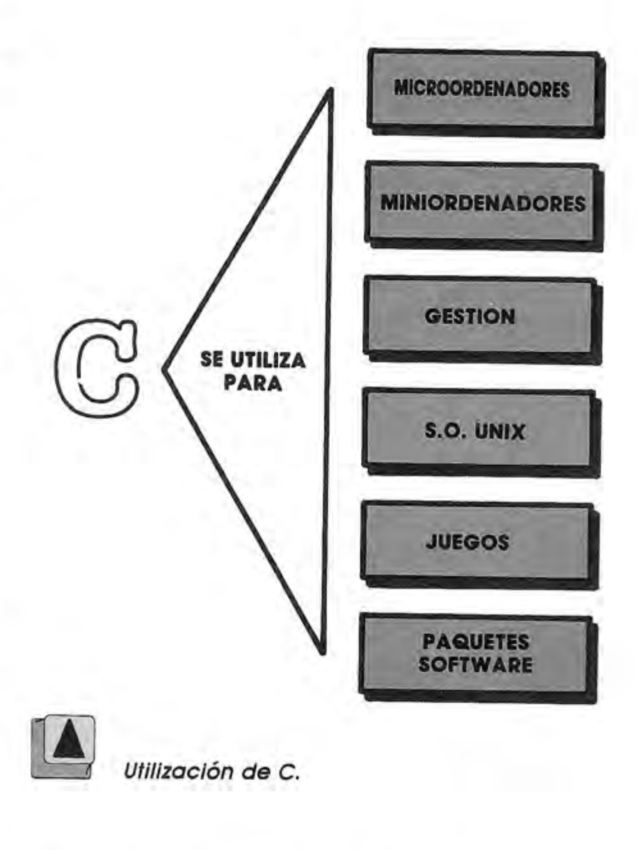

C reduce los gastos de instalación.




C es una buena herramienta para el diseño.




C se utiliza para paquetes Software y juegos.




Utilización de C.

La escritura de programas en C y su compilación

Los lenguajes de programación, en general, están basados en lenguajes "intérpretes" y lenguajes "compilados". Entre los primeros podemos citar el LOGO y el BASIC, perteneciendo al segundo grupo de los llamados compilados lenguajes tales como el FORTRAN, PASCAL y C.

La primera fase para escribir un programa en C es disponer de un editor de propósito general, ya que el C no dispone de uno propio, para empezar a escribir su programa. Su ordenador dispondrá de un editor para realizar esta tarea. Si prefiere utilizar un editor distinto del que dispone su sistema, puede recurrir a procesadores de texto tan conocidos como el Wordstar utilizando la opción **N** (opción de "no documento"). Una vez escrito su programa, debe darle un nombre recordando que debe terminar con **c**, como, por ejemplo:

```
prog.c
hola.c
```

En un sistema UNIX el editor podría ser invocado tecleando **ed, ex, edit, emacs,** o **vi.** Los ordenadores personales dispo-

nen de editores tales como **edlin, ed, Wolkswriter,** etc.

Una vez escrito su programa, éste debe compilarse por medio del compilador de C, cuya misión es detectar si su programa posee algún error y, en caso de no existir ninguno, traducirlo al lenguaje que entiende la máquina, es decir, el código objeto. La traducción de su programa lo colocará en un nuevo fichero. Bastará invocar el nombre de este último fichero para que nuestro programa se ejecute.

Veamos un ejemplo:

```
# include <stdio.h>
main ( )
{
printf ("Este es mi primer programa en C"\n);
}
```

Este texto que hemos tecleado, gracias al editor, se guardaría en un fichero y se compilaría dando como resultado un fichero que al ser invocado ejecutaría nuestro programa.

En el sistema UNIX, el compilador de C se denomina **cc,** por lo que para compilar nuestro programa bastaría teclear:

```
cc nombre de mi programa
```

Terminada la compilación observaríamos que se ha creado un nuevo fichero llamado **a.out.** Tecleando dicho nombre nuestro programa se ejecutaría apareciendo en pantalla el siguiente texto:

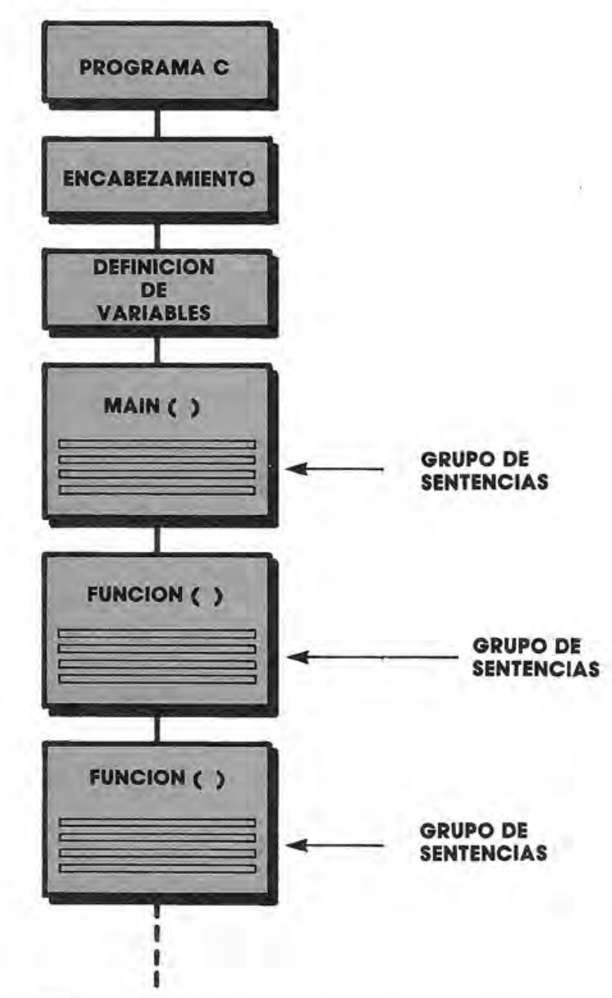
Este es mi primer programa en C. ¡Era de esperar!

Para ordenadores personales existe una versión de compilador de C llamada Laticce C. Los pasos que daremos para ejecutar nuestro programa serán los siguientes:

```
mc1 nombre de mi programa
mc2 nombre de mi programa
link c nombre de mi programa
```

Este último paso originará un fichero llamado **nombre de mi programa.exe** y si a continuación tecleamos **nombre de mi programa.exe** conseguiremos ejecutar nuestro programa.

La estructura en general de un programa C sería:





Elementos del lenguaje C

El lenguaje C soporta los siguientes tipos de datos:

- Caracteres.
- Enteros.
- Números en coma flotante.
- Funciones.
- Arrays y punteros.
- Estructuras.
- Uniones.

Operadores tales como:

- Aritméticos.
- Relacionales.
- Lógicos.
- Manejo de bits.
- Asignación.
- Condicional.
- Acceso a datos.

Estructuras de control tales como:

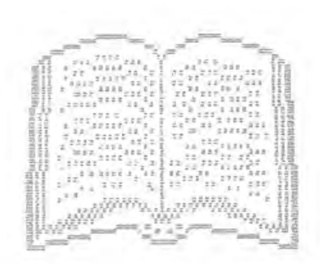
— Sentencias.
— Sentencia «If-Then-Else».
— Sentencia «switch».
— Saltos y etiquetas.

Tipos de constantes tales como:

— Enteros: dígitos numéricos.
— Números en coma flotante (caracteres y constantes alfanuméricas).

Funciones de entrada/salida, que forman parte de la librería estándar del C, y entre otras citaremos:

— getchar.
— printf.
— etc.

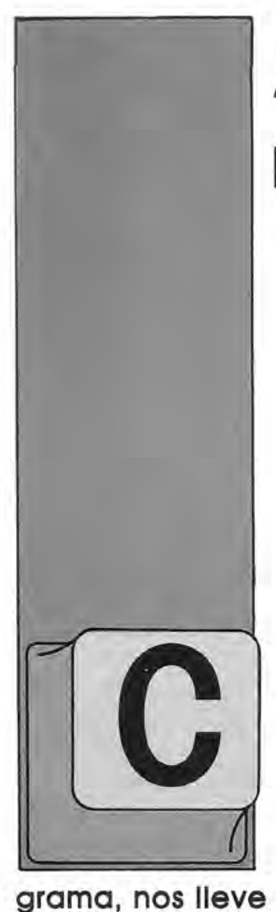




C posee una librería estándar.

APLICACIONES

INTRODUCCION



CUANDO nos enfrentamos con un problema que debemos resolver mediante un ordenador de modo inmediato, pensamos en un algoritmo que, traducido en un programa, nos lleve a la solución. Pero esta forma de enfocar la resolución de problemas mediante ordenador presenta dos inconvenientes graves para un usuario de tipo medio:

— Por cada problema distinto a resolver deberíamos tener un programa que nos lo solucionara, con la consiguiente acumulación de programas de funcionamiento y manejo dispar.

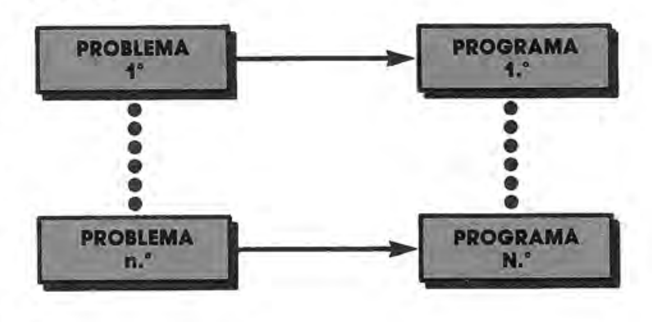
— Muchas veces la inexistencia de programas específicos para una serie de problemas determinados lleva consigo que el usuario deba crear por sí mismo o encargar a una empresa especializada un programa para resolverlo.

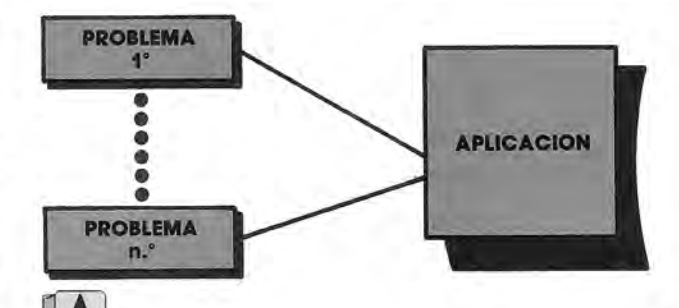
Normalmente el usuario no es un experto en Informática, sino que, simplemente, debe utilizar un ordenador en su actividad diaria para resolver una serie de problemas.

Por ello, lo que en estos casos debe preocupar y ocupar su tiempo es en resolver el problema, no en cómo decirle al ordenador cómo resolverlo.

Para ello, desde hace algún tiempo están apareciendo en el mercado una serie de programas de uso general, que llamaremos aplicaciones, que permiten concentrarse en la resolución del problema sin necesidad de programar el ordenador, ni de conocimientos específicos sobre Informática.

Estas aplicaciones están diseñadas de tal forma que con ellas es posible abarcar todo el espectro de necesidades más usuales en la actividad diaria con un ordenador, eludiendo la mayoría de las veces el uso de programas específicos. De hecho un usuario no tiene por qué utilizar más que tres o cuatro programas de uso general.





Las aplicaciones permiten resolver más de un problema.



Tipos de aplicaciones

Dentro del conjunto de aplicaciones disponibles en el mercado podemos hacer la siguiente división:



Procesadores de texto

Estos programas nos permiten crear y editar textos, como cartas, memorándums, etc.

De hecho, transforman un ordenador en una máquina de escribir muy sofisticada, con unas posibilidades impensables en cualquier otro tipo de máquina de escribir. Los procesadores de texto nos permiten componer, poner márgenes, corregir de forma interactiva el texto y un montón más de posibilidades de edición.

Con un procesador de textos pueden crearse una serie de cartas personalizadas con la ayuda de una base de datos, para así realizar una circular a unos clientes, por ejemplo.

De entre los procesadores de texto, sin duda el más conocido es el WORDSTAR, del que hay innumerables versiones para casi todos los ordenadores.

Otros procesadores de texto muy conocidos son:

— LOCOSCRIPT.
— MICROSOFT WORD.
— WRITING ASSISTANT.
— PERSONAL EDITOR.
— EASY WRITER.
— WORD PERFECT.



Hojas de cálculo

Las llamadas "hojas de cálculo" son unas aplicaciones que permiten realizar de forma sencilla cualquier tipo de cálculo en el cual intervengan tablas de números y relaciones numéricas entre ellos. Su propiedad fundamental es la de la "actualización automática", que consiste en que si cambiamos el valor de un número todos los valores y fórmulas relacionados con él se actualizarán, de forma automática, permitiendo con ello canalizar situaciones y proyectos, modelos de acumulación de situaciones reales.

Su uso más frecuente consiste en cálculos financieros y estimación de tendencias, pero con las hojas de cálculo actuales podemos resolver casi cualquier problema de cálculo, llegando a

B:LPALICO PAG. 1 LIN. 1 COL 01 INSERTAR Sí

MENU PRINCIPAL

Movimiento de Cursor	—Borrar—	—Varios—	—Otros Menús—
^S car. izda. ^D car. dcha.	^G car.	^Tab. ^B Recomp.	(sólo desde Pral)
^A pal. izda. ^F pal. dcha.	DEL car iz	^V INSERTAR SI/NO	^J Ayuda
^E lin. arr. ^X lín. abajo	^T pal dch	^L Bus./sust. otra	^Q Rápido
— Deslizar: —	^Y línea	RETORNO fin párrafo	^O Pantalla
^Z línea abajo ^W línea arr		^N Insertar RETORNO	^K Bloques
^C pant. arr ^R pant. abajo		^U Cancelar comando	^P Impresión

PROGRAMA DE APLICACIONES

— Introducción a los programas de aplicación.
— Tipos y usos de los principales programas.
— Tratamiento de textos: WordStar.
— Hojas de cálculo: Multiplán, Lotus 1-2-3.
— Bases de datos: dBase II, dBase III.
— Paquetes integrados: Open Access, Symphony, Framework.

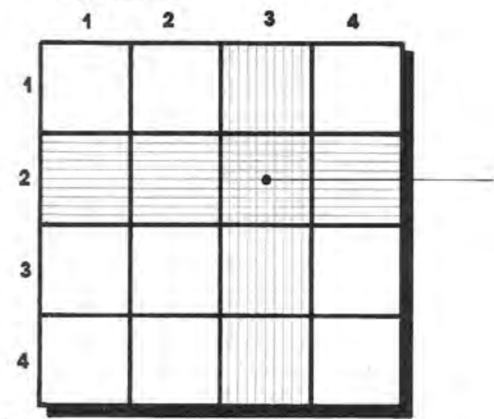
1CENTRA 2PON MI 3PON MD 4FIGTAB 5BORTAB 6RECOMP 7VER 8SUBRAY 9NEGR 10Más



Ejemplo de procesador de textos: WordStar.

convertirse en verdaderos "lenguajes" de programación.

La idea básica de una hoja de cálculo es la siguiente:




Concepto de hoja de cálculo.

Tenemos una matriz bidimensional en la cual cada intersección de fila y columna es lo que se llama una "celda", en la cual puede almacenarse un valor numérico o una fórmula que esté relacionada con otras celdas. De este modo podemos llegar a establecer una serie de relaciones entre las celdas que nos permiten realizar los cálculos que precisemos.

En el mercado hay muchas hojas de cálculo, de entre las cuales las más extendidas son:

— VISICALC.
— MULTIPLAN.
— LOTUS 1-2-3.
— SUPERCALC.
— TK SOLVER! (para aplicaciones científicas).



Bases de Datos

Una de las utilizaciones más comunes de un ordenador es la de almacenamiento y recuperación de grandes cantidades de datos, ordenados y estructurados de alguna manera determinada.

Para este tipo de aplicaciones existen en el mercado las llamadas "bases de datos", que son programas para gestionar graves soluciones de información de tal forma que podamos almacenarlos y recuperarlos de una forma coherente, práctica y eficaz.

Normalmente, los datos están almacenados en algún tipo de dispositivo físico, siendo el más común el disco flexible, aunque es recomendable el uso de discos duros de tecnología Winchester por su mayor velocidad de acceso.

Las bases de datos nos permiten guardar y recuperar los datos de una forma determinada por medio de sistemas es-

	1	2	3	4	5	6	7
1		VENTA DE ELEMENTOS VARIOS EN LOS PRIMEROS 6 MESES					
2		ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO
3							
4	ITEM#1	132	23	214	45	2342	14
5	ITEM#2	1231	56	35	456	23	386
6	ITEM#3	345	457	2345	3456	234	368
7	ITEM#4	5667	23	6	36	234	0
8	ITEM#5	343	568	678	234	234	52
9	ITEM#6	343	123	12	343	234	246
10	ITEM#7	43	6789	68	568	234	211715
11	ITEM#8	97	24	28	457	234	28
12	ITEM#9	23	7878	13	234356	353345	789
13	ITEM#10	42221	776	7	234	66	123
14	ITEM#11	333	67325	79	234	55	456
15	ITEM#12	113	344	124	234	5	4645
16							
17	TOTAL	50891	84386	3604	240653	357240	218830
18							

MANDATO: Alfa Blan Clasif Direc Edit Format Genera HoAyu Imprimir Limit
Mov Nomb Opcion Proteg Quitar Replic Salir Transf Valor Xterno
Seleccionar opción o pulsar inicial de mandato
R1C2 "VENTA DE ELEMENTOS VARIOS E 97% Libre NL Multiplan: TEMP


Ejemplo de hoja de cálculo. Multiplán.

APELLIDO	DIRECCION	CIUDAD	PROVINCIA	CP	TELEFONO
Nieva	Ercilla, 10	Bilbao	Vizcaya	48025	(94) 447 09 87
Lemos	Ayala, 96	Madrid	Madrid	28001	(91) 250 50 13
Barroso	Lucena, 32	Ronda	Málaga	29033	(952) 32 45 91
Ruano	Somera, 88	Rivadavia	Orense	32017	(988) 47 62 90
Romero	Tutor, 41	Linares	Jaén	23010	(953) 13 42 58
Gutiérrez	Princesa, 27	Cuenca	Cuenca	16016	(966) 57 83 49
Gómez	Gran Vía, 81	Cuéllar	Segovia	40020	(911) 84 65 92
Cueto	Estrella, 1	Lérida	Lérida	25041	(973) 38 97 62
Cuevas	Diagonal, 92	Mataró	Barcelona	08005	(93) 256 47 56
Lucena	Torrijos, 5	Toledo	Toledo	45009	(925) 34 78 91



Ejemplo de base de datos = dBase III.

peciales de ordenación de ficheros, y recuperar toda o una parte de la información de una forma muy sencilla.

Sin duda, la base de datos más conocida es la dBASE III, estando su uso muy extendido para aplicaciones de tamaño medio. Para aplicaciones con poco volumen de datos existen otros programas como el PFS:FILE o el FILLING ASSISTANT, que permiten manejar los datos de una forma muy sencilla.



Paquetes integrados

Cuando necesitamos realizar distintas operaciones sobre un mismo conjunto de datos, como cálculos, actualizaciones de una base de datos, necesitamos poder intercambiar los datos entre una aplicación y otra con el problema de que muchas veces el formato de los datos de las aplicaciones que manejamos no son compatibles entre sí. Por ello se han diseñado los llamados paquetes integrados, que no son más que una reunión de todos los programas que hemos visto en una sola aplicación. Esto es, un paquete que reúne en una sola aplicación un procesador de texto, una hoja de cálculo, una base de datos, etc.

Estos paquetes integrados permiten resolver cualquier tipo de problema sin necesidad de recurrir a otros programas.

De entre los paquetes integrados más conocidos podemos citar:

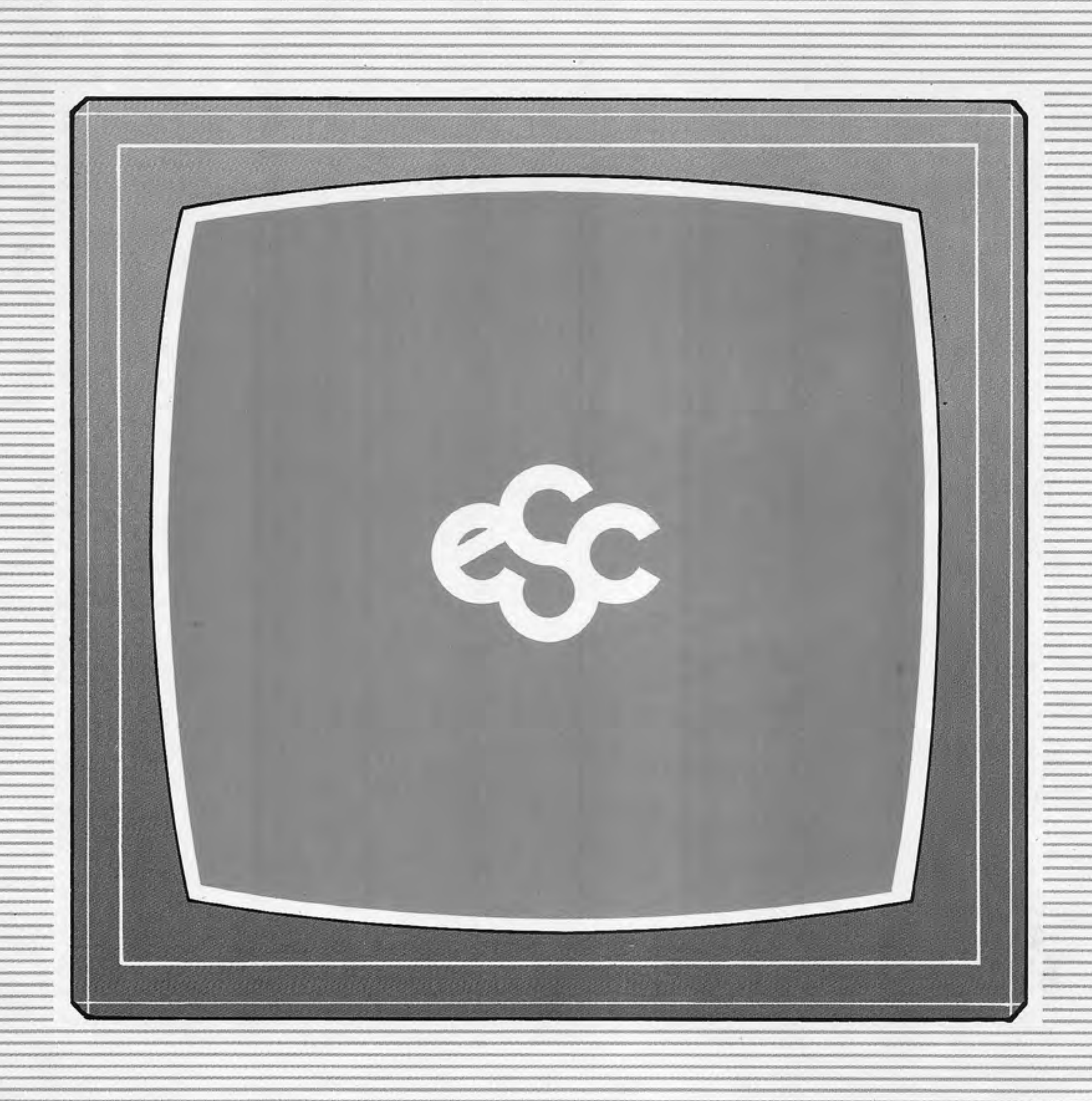
— SYMPHONY.
— OPEN ACCESS.
— FRAMEWORK.

```
Disco  Crear  Edit  Local  Ventana  Texto   Números  Gráfico  Imprimir  0:11:19
 (HOJA DE CALCULO)
            A                          B    en despacho, gafas de sol y un
 1                                 Saldo    ayudarle a conseguir las dos
 2   Depósito anual          2.000 Ptas.    ndrá de un magnífico despac    <A:>
 3   Tasa de interés             11,00 %    work).                         <B:>
 4   Número de depósitos              20                                   <C:>
 5                                          este rectángulo para que       <D:>
 6   Tasa de impuestos           30,00 %    ravés suyo:                    <(:>
 7
 8   Valor futuro sin impuestos  128.406 Pt.
 9   Valor futuro con impuestos   61.977 Pt.
 10
 11  Sin imp. menos imp.          66.429 Pt.   e "Q" ahora.

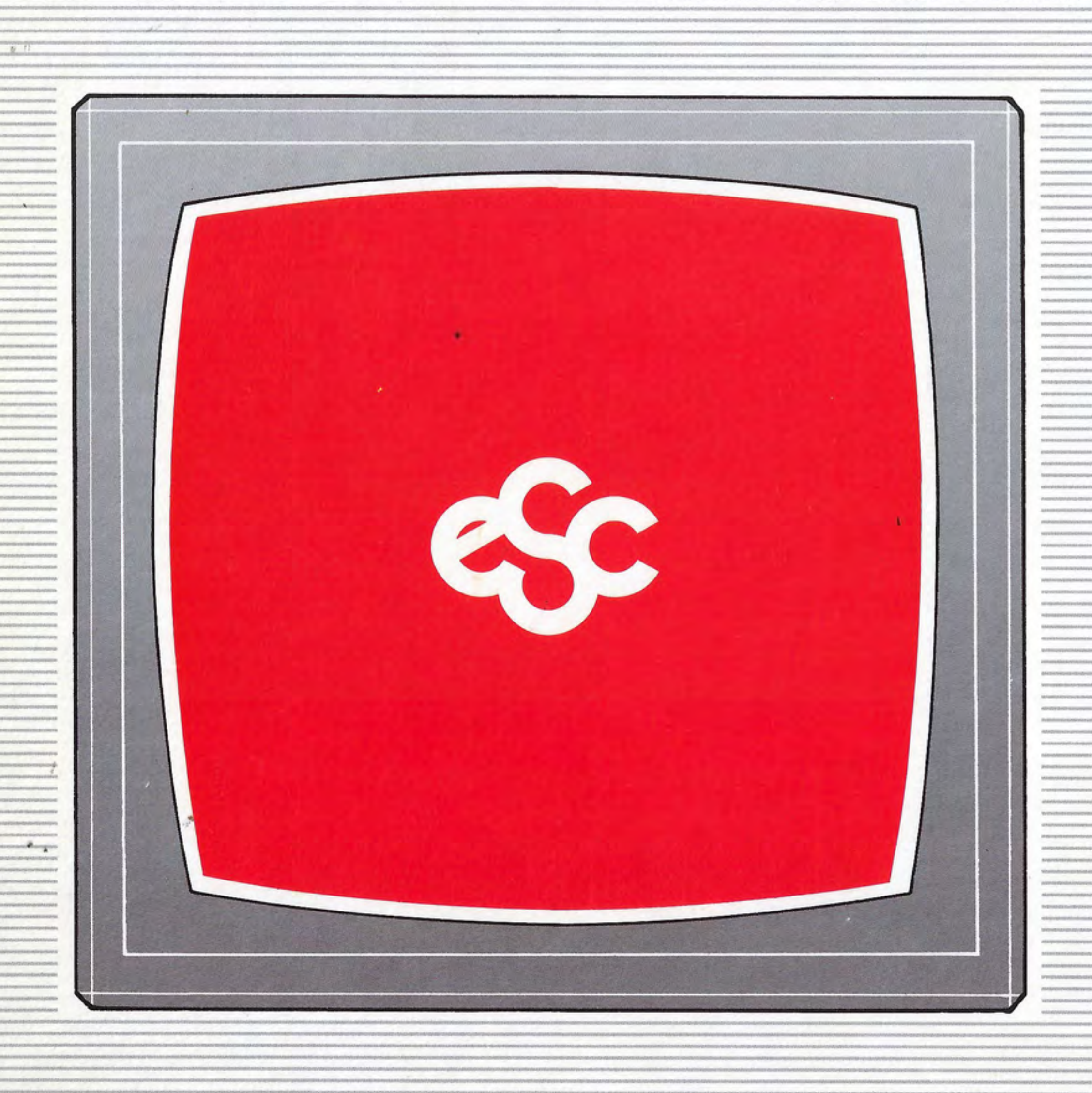
 (DataBase)
   AGENTE   DIVISION   SECTOR   SITUACION   APODO   ESPECIA

   Q17      Parcelas   52       Berlín      Spike   Gestión      (DataBase)
                                                                 (HOJA DE)
                                                                 (PROCESO)
          |     HOJA DE CALCULO     | Doc:      2/3          MAY
```

Ejemplo de paquete integrado. Framework.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼